HoGent
BUSINESS AND INFORMATION MANAGEMENT

Professional Bachelor in Applied Computer Science
Academic year 2012-2013

Solving CAPTCHA using neural networks

Submitted on 10 June 2013

Student:
Pieter Van Eeckhout

Mentor:
Johan Van Schoor

HoGent Business & Information Management
Professional Bachelor in Applied Computer Science
Academic year 2012-2013

Solving CAPTCHA using neural networks

Submitted on 10 June 2013

Student:
Pieter Van Eeckhout

Mentor:
Johan Van Schoor

Contents

Abstract

Preamble

Firstly, dear reader, I would like to thank you for taking the time to read this thesis. Without an audience this entire endeavour would not mean as much as it does right now, while you are reading it's results. I personally believe this is because I would like my life not to go unnoticed. So if this thesis helps, or influences you in any way, then this work has gained more meaning.
Secondly I would like to thank the following persons who have made it possible for me to arrive at this point. Special thanks and mentions go to:

- my parents, for supporting me and giving me the opportunity and supplying the means for me to pursue my academic career.
- my girlfriend, because she has helped me countless times, she helped me through the rough spots. Because she never once complained about the time consuming job of writing this work.
- my good friends, willing proof readers and content critics Wouter Dekens, Patrick Van Brussel and Thijs van der Burgt.
- Johan Van Schoor and Bert Van Vreckem for the support, organisation, guidance and feedback.

Bare in mind that this is not an exclusive list. So lastly I would like to thank all the other people who are not mentioned by name, like the teaching and support staff at University College Ghent.

Ghent BELGIUM, June 2013

[illegible]

Pieter Van Eeckhout

Chapter 1

Solving CAPTCHA using neural networks

The target audience. This thesis was written with an audience in mind that already has some technical understanding of computers and how they operate on hardware level (processor etc.). If you feel that your current knowledge is insufficient, or just want to read up some more, then I refer you to the "How Computers Work - Processor and Main Memory" [Young, 2001] e-book.

The history of SPAM. Ever since the internet found its way into our daily lives, there have been people out there who don't always have other people's best interests in mind. I am referring to spammer, people aiming to advertise their product, services, etc . . . in an aggressive manner. The methods of advertising include but are not limited to:

- Sending bulk emails without the recipients permission (SPAM).
- Posting irrelevant links and information on fora and various social media.
- Flooding chat channels with their links and information.

These emails, posts and messages inconvenience the end-users, requiring time to filter out the junk. The economic costs of SPAM has led to a decrease in the Japanese GDP by 500 billion Yen (3.78 billion Euro) in 2004 and were projected to reach a decrease of 1% of the total GDP by 2010 unless adequate countermeasures were taken [Ukai and Takemura, 2007]. [Khong, 2004] researched the economic arguments for regulating junk mails and the efficiency of these regulations.

Birth of CAPTCHA. The two previously mentioned researches signify the importance and impact of SPAM on our daily lives. The users of the internet quickly tried to implement methods to prevent spammers from spreading their advertisements to the masses. Several prevention and detection methods and systems were developed successfully. These range from hidden text only visible to automated scripts, to invalid HTML tags. One of the methods developed for this purpose is a CAPTCHA test. CAPTCHA is an acronym based on the word "capture" and stands for 'Completely Automated Public Turing test to tell Computers and Humans Apart'. An attempt to trademark the term was made by Carnegie Mellon University on 15 October 2004, but the application was eventually dropped on 12 April 2008

Spammers fight back. All these prevention and detection methods did not stop the spammers from trying to reach an audience as large as possible. The spammers rely on a large target audience because of the return rates being as low as 0.0023% [Cobb, 2003]. Trying to reach such a large audience the spammers start to device ways to circumvent or break the existing systems. One of these methods is solving CAPTCHA tests by making use of the adaptive learning and pattern recognizing capabilities of neural networks. These networks can be used to recognize letters from images with adversarial clutter. This is the area I will focus on in this thesis. This thesis will list some of the difficulties regarding the extraction of relevant data from a CAPTCHA, and how to possibly overcome these difficulties. However the main focus will be on searching for the types and configuration of neural networks best used for pattern recognition.

Chapter 2

Premise and research questions

2.1 Premise

The main objective of this thesis is to ascertain whether neural networks are capable of solving the current generation of CAPTCHA images. we will define the premise as following:

"Are neural networks a viable tool for solving the current generation of CAPTCHA?"

2.2 Research questions

The research can be divided into two separate subjects. If one was to develop software for automatic CAPTCHA solving, following questions and problems would need to be addressed.

CAPTCHA:

- What are the different types of CAPTCHA?
- How can the distorted text be extracted?

Neural networks:

- How do neural networks operate?
- Which types of neural networks are well suited pattern recognition?
- What network configuration would perform best?

general:

- How future proof would this solution be?
- Is there enough economic incentive to invest in development?

Chapter 3

Methodology

Research philosophy.

Research approach.

Access.

Research strategy.

Chapter 4

Corpus

4.1 CAPTCHA

4.2 Neural Networks

4.3 Implementation

Chapter 5

Conclusion

Appendix A

Sourcecode

A.1 Package be

A.2 Package be.hogent

A.3 Package be.hogent.bulksolvingstatistics

A.4 Package be.hogent.captchabuilder

A.5 Package be.hogent.captchacleanup

A.6 Package be.hogent.captchasolvingnetwork

Listing A.1: be.hogent.bulksolvingstatistics.BulkSolvingStatistics

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.bulksolvingstatistics;

import be.hogent.bulksolvingstatistics.domain.DomainFacade;
import be.hogent.bulksolvingstatistics.persistance.PersistanceController;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * BulkSolvingStatistics.java (UTF-8)
 *
 * This Is the main startup class for the neural network based captcha
     solving
 * system used for generating the statistical results used in my thesis.
 *
 * 2013/05/19
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class BulkSolvingStatistics {

    private static BulkSolvingStatistics instance;
    private DomainFacade domain;
    private PersistanceController persistance;

    /**
     * Program startup method, calls upon it's constructor.
     *
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        instance = new BulkSolvingStatistics();

    }

    /**
     * Default constructor.
     *
     */
    public BulkSolvingStatistics() {
        try {
            persistance = PersistanceController.getInstance();
            domain = new DomainFacade(persistance);
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(BulkSolvingStatistics.class.getName()).log(
                Level.SEVERE, null, ex);
        } catch (SQLException ex) {
            Logger.getLogger(BulkSolvingStatistics.class.getName()).log(
                Level.SEVERE, null, ex);
        }
    }
}
```

A.7 Package be.hogent.bulksolvingstatistics.domain

A.8 Package be.hogent.bulksolvingstatistics.persistance

A.9 Package be.hogent.bulksolvingstatistics.ui

A.10 Package be.hogent.captchabuilder.builder

A.11 Package be.hogent.captchabuilder.elementcreator

A.12 Package be.hogent.captchabuilder.util

Listing A.2: be.hogent.captchacleanup.CaptchaCleanup

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package be.hogent.captchacleanup;

import be.hogent.captchacleanup.utils.textfromimage.GetImageText;
import java.awt.image.BufferedImage;
import java.util.LinkedList;

/**
 *
 * @author Pieter
 */
public class CaptchaCleanup {

    public static BufferedImage drawBoxesAroundText(BufferedImage image) {
        GetImageText myget = new GetImageText(image);

        LinkedList boxes = myget.getTextBoxes();
        return myget.isolateText(boxes);
    }
}
```

A.13 Package be.hogent.captchacleanup.utils

Listing A.3: be.hogent.captchasolvingnetwork.CaptchaSolvingNetwork

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchasolvingnetwork;

import be.hogent.bulksolvingstatistics.domain.neuralnetwork.encogutils.
    EncogTrainingSet;
import be.hogent.captchabuilder.util.ArrayUtil;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import be.hogent.captchasolvingnetwork.network.encog.EncogHopfieldNetwork;
import be.hogent.captchasolvingnetwork.network.encog.
    EncogHopfieldNetworkBuilder;
import java.util.Arrays;
import org.apache.log4j.Logger;

/**
 * CaptchaSolvingNetwork.java (UTF-8)
 *
 * This Is the main startup class for the neural network based captcha
     solving
 * system. This system will call the captcha-cleanup library and will pass
     the
 * results of the image cleanup to the neural networks.
 *
 * 2013/04/28
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.1.0
 */
public class CaptchaSolvingNetwork {

    private static final Logger logger;

    static {
        logger = Logger.getLogger(CaptchaSolvingNetwork.class);
    }

    public static void main(String[] args) {
        new CaptchaSolvingNetwork();
    }

    public CaptchaSolvingNetwork() {
        this(ArrayUtil.concat(CaptchaConstants.LETTERS, CaptchaConstants.
            NUMBERS, CaptchaConstants.SPECIAL), 40, 50);
    }

    public CaptchaSolvingNetwork(char[] chars, int hSize, int vSize) {
        chars = new char[] {'a', '5'};
        double[] input, result, expectedResult;
        double[][] inputTrainingsSet = EncogTrainingSet.
            buildTrainingInputSet(chars, hSize, vSize);
        double[][] outputTrainingsSet = EncogTrainingSet.
            buildTrainingIdealSet(chars);


        System.out.println("creating,␣building␣and␣traing␣hopfield␣network")
            ;
        EncogHopfieldNetwork hopfield = new EncogHopfieldNetworkBuilder(
            inputTrainingsSet, hSize, vSize).createEncogHopfieldNetwork();
        hopfield.buildNetwork();
        hopfield.trainNetwork();
        //evaluate over trainingset
        System.out.println("Evaluating␣hopfield␣network␣over␣trainingset");
        for (int i = 0; i < inputTrainingsSet.length; i++) {
            input = inputTrainingsSet[i];
            expectedResult = outputTrainingsSet[i];
            result = hopfield.evaluate(input, 100);

            if (Arrays.equals(result, expectedResult)) {
                System.out.println(chars[i] + "␣recognized␣correctly");
            } else {
                System.out.println(chars[i] + "␣recognized␣Incorrectly");
                System.err.println("result:␣" + Arrays.toString(result) + "␣
                    !=␣" + Arrays.toString(expectedResult));
            }
        }
    }
}
```

A.14 Package be.hogent.captchasolvingnetwork.encog$_2$

A.15 Package be.hogent.captchasolvingnetwork.network

A.16 Package be.hogent.captchasolvingnetwork.util

Listing A.4: be.hogent.bulksolvingstatistics.domain.DomainFacade

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.bulksolvingstatistics.domain;

import be.hogent.bulksolvingstatistics.domain.neuralnetwork.
    DefaultNeuralNetworkController;
import be.hogent.bulksolvingstatistics.domain.neuralnetwork.
    DefaultNeuralNetworkRepository;
import be.hogent.bulksolvingstatistics.domain.neuralnetwork.
    NeuralNetworkController;
import be.hogent.bulksolvingstatistics.domain.neuralnetwork.
    NeuralNetworkRepository;
import be.hogent.bulksolvingstatistics.domain.neuralnetwork.dataobjects.
    NeuralNetworkDataObjectBuilder;
import be.hogent.bulksolvingstatistics.domain.neuralnetwork.encogutils.
    EncogTrainingSet;
import be.hogent.bulksolvingstatistics.persistance.PersistanceController;
import be.hogent.captchabuilder.util.ArrayUtil;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import be.hogent.captchasolvingnetwork.network.encog.EncogBasicNetwork;
import be.hogent.captchasolvingnetwork.network.encog.
    EncogBasicNetworkBuilder;
```

```
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * DomainFacade.java (UTF-8)
 *
 * Acts as the entrypoint for the domain layer. All calls toward the domain
 * should pass here. Will contain the repositories and the controller
    instances,
 * if there are any.
 *
 * 2013/05/19
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class DomainFacade {

    public NeuralNetworkRepository networkRepository;
    public NeuralNetworkController networkController;
    private NeuralNetworkDataObjectBuilder builder;

    public DomainFacade(PersistanceController persistance) {
        this(persistance, new DefaultNeuralNetworkRepository(), new
            DefaultNeuralNetworkController());
    }

    public DomainFacade(PersistanceController persistance,
        NeuralNetworkRepository networkRepository, NeuralNetworkController
        networkController) {
        this.networkRepository = networkRepository;
        this.networkController = networkController;

        System.out.println("building charset");
        char[] chars = ArrayUtil.concat(CaptchaConstants.LETTERS,
            CaptchaConstants.NUMBERS, CaptchaConstants.SPECIAL);
        bulkTest(chars, 40, 50);
    }

    private void bulkTest(char[] chars, int hSize, int vSize) {
        System.out.println("creating training input");
        double[][] inputTrainingsSet = EncogTrainingSet.
            buildTrainingInputSet(chars, hSize, vSize);
        System.out.println("building training ideal");
        double[][] outputTrainingsSet = EncogTrainingSet.
            buildTrainingIdealSet(chars);


        for (double accuracy = 0.01; accuracy > 0.0001; accuracy -= 0.00005)
             {
            for (int hiddenLayerSize = 1000; hiddenLayerSize <= 4000;
                hiddenLayerSize += 1000) {
                int[] hiddenlayers = new int[]{hiddenLayerSize};
                EncogBasicNetwork network = new EncogBasicNetworkBuilder(
                    inputTrainingsSet, outputTrainingsSet)
                        .setHiddenLayers(hiddenlayers)
                        .setAccuracy(accuracy)
                        .createEncogBasicLetterRecognitionNetwork();
                network.buildNetwork();

                long startTimeLong = System.nanoTime();
                network.trainNetwork();
                long endTimeLong = System.nanoTime();
                double durationInSec = (double) ((endTimeLong -
                    startTimeLong) / Math.pow(10, 9));


                builder = new NeuralNetworkDataObjectBuilder();
                builder.setNetworkType("basic")
                        .setLayerLayout(network.getLayerLayout())
                        .setAccuracy(accuracy)
                        .setTrainingDuration(durationInSec)
                        .setIterations(vSize)
                        .setSavedLocation("");
                try {
                    //save the network and set the ID
                   network.setId(PersistanceController.getInstance().
                       addNetwork(builder.createNeuralNetworkDataObject()).
                       getId());
                } catch (SQLException | ClassNotFoundException ex) {
                    Logger.getLogger(DomainFacade.class.getName()).log(Level
                        .SEVERE, null, ex);
                    System.exit(1);
                }

                networkController.setNetwork(network);
                networkController.evaluate(":TEXT!TEXTPRODUCER#
                    ALPHANUMERIC_SPECIAL@MINLENGTH*1@MAXLENGTH*1", 50, 100);
            }
        }
    }
}
```

A.17 Package be.hogent.bulksolvingstatistics.domain.neuralnetw

Listing A.5: be.hogent.bulksolvingstatistics.persistance.DatabaseConnection

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.bulksolvingstatistics.persistance;

import java.sql.*;

/**
 * DatabaseConnection.java (UTF-8)
 *
 * This class maintains the connection between the application and the
     SQLite
 * database.
 *
 * 2013/05/20
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class DatabaseConnection {

    private final static String JDBC = "org.sqlite.JDBC",
            SQLITEPATH = "jdbc:sqlite:DataBase/BulkStatistics";
    private Connection connection;

    /**
     * Default constructor for DatabaseConnection
     *
     * @throws ClassNotFoundException
     * @throws SQLException
     */
    public DatabaseConnection() throws ClassNotFoundException, SQLException
        {
        Class.forName(JDBC);
        connection = DriverManager.getConnection(SQLITEPATH);
    }

    /**
     * This method closes the connection between the application and the
     * database. This primary to reduce the memory cost.
     *
     * @throws SQLException
     */
    public void closeConnection() throws SQLException {
        connection.close();
    }

    /**
     * This method return an existing connection. Otherwise it creates a new
     * one.
     *
     * @return
     * @throws SQLException
     * @throws ClassNotFoundException
     */
    public Connection getConnection() throws SQLException,
        ClassNotFoundException {
        if (connection == null || connection.isClosed()) {
            Class.forName(JDBC);
            connection = DriverManager.getConnection(SQLITEPATH);
        }
        return this.connection;
    }
}
```

A.18 Package be.hogent.bulksolvingstatistics.persistance.mappe

Listing A.6: be.hogent.bulksolvingstatistics.persistance.PersistanceController

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.bulksolvingstatistics.persistance;

import be.hogent.bulksolvingstatistics.domain.neuralnetwork.dataobjects.
    NeuralNetworkDataObject;
import be.hogent.bulksolvingstatistics.domain.neuralnetwork.dataobjects.
    TestResultDataObject;
import be.hogent.bulksolvingstatistics.persistance.mappers.
    NeuralNetworkMapper;
```

```
import be.hogent.bulksolvingstatistics.persistance.mappers.TestResultMapper;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.Collection;

/**
 * PersistanceController.java (UTF-8)
 *
 * Acts as the entrypoint for the persistence layer. All calls toward the
 * Databases should pass here.
 *
 * 2013/05/19
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class PersistanceController {

    private static PersistanceController persistanceController;
    private NeuralNetworkMapper networkMapper;
    private TestResultMapper testResultMapper;
    private DatabaseConnection connection;

    /**
     * This method creates a singleton instance of PersistanceController.
     *
     * @return the singleton instance of PersistanceController
     * @throws ClassNotFoundException
     * @throws SQLException
     */
    public static PersistanceController getInstance() throws
        ClassNotFoundException, SQLException {
        if (persistanceController == null) {
            persistanceController = new PersistanceController();
        }
        return persistanceController;
    }

    private PersistanceController() throws ClassNotFoundException,
        SQLException {
        connection = new DatabaseConnection();
        networkMapper = new NeuralNetworkMapper();
        testResultMapper = new TestResultMapper();
    }

    /**
     * Passes the CRUB operation to the mapper
     *
     * @param network NeuralNetworkDataObject to be saved
     * @return a NeuralNetworkDataObject containing the data
     * @throws SQLException
     * @throws ClassNotFoundException
     * @see NeuralNetworkMapper
     */
    public NeuralNetworkDataObject addNetwork(NeuralNetworkDataObject
        network) throws SQLException, ClassNotFoundException {
        return networkMapper.add(network);
    }
    /**
     * Passes the CRUB operation to the mapper
     *
     * @return a Collection holding all the a NeuralNetworkDataObjects in
     * database
     * @throws ClassNotFoundException
     * @throws SQLException
     */
    public Collection<NeuralNetworkDataObject> findAllNetworks() throws
        ClassNotFoundException, SQLException {
        return networkMapper.findAll();
    }

    /**
     * Passes the CRUB operation to the mapper
     *
     * @param id the id of the NeuralNetworkDataObject to be loaded.
     * @return a NeuralNetworkDataObject containing the data
     * @throws SQLException
     * @throws ClassNotFoundException
     * @see NeuralNetworkMapper
     */
    public NeuralNetworkDataObject findNetwork(int id) throws
        ClassNotFoundException, SQLException {
        return networkMapper.find(id);
    }

    /**
     * Passes the CRUB operation to the mapper
     *
     * @param network NeuralNetworkDataObject to be updated
     * @return a NeuralNetworkDataObject containing the data
     * @throws SQLException
     * @throws ClassNotFoundException
     * @see NeuralNetworkMapper
     */
    public NeuralNetworkDataObject updateNetwork(NeuralNetworkDataObject
        network) throws SQLException, ClassNotFoundException {
        return networkMapper.upate(network);
    }

    /**
     * Passes the CRUB operation to the mapper
     *
     * @param network NeuralNetworkDataObject to be removed
     * @throws SQLException
     * @throws ClassNotFoundException
     * @see NeuralNetworkMapper
     */
    public void removeNetwork(NeuralNetworkDataObject network) throws
        SQLException, ClassNotFoundException {
        networkMapper.delete(network);
    }

    /**
     * Passes the CRUB operation to the mapper
     *
     * @param testResult TestResultDataObject to be saved
     * @return a TestResultDataObject containing the data
     * @throws SQLException
     * @throws ClassNotFoundException
     * @see TestResultMapper
     */
    public TestResultDataObject addTestResult(TestResultDataObject
        testResult) throws SQLException, ClassNotFoundException {
        return testResultMapper.add(testResult);
    }

    /**
     * Passes the CRUB operation to the mapper
     *
     * @return a Collection holding all the a TestResultDataObject in
       database
     * @throws SQLException
     * @throws ClassNotFoundException
     * @see TestResultMapper
     */
    public Collection<TestResultDataObject> findAllTestResults() throws
        ClassNotFoundException, SQLException {
        return testResultMapper.findAll();
    }

    /**
     * Passes the CRUB operation to the mapper
     *
     * @param id the id of the TestResultDataObject to be found
     * @return a TestResultDataObject containing the data
     * @throws SQLException
     * @throws ClassNotFoundException
     * @see TestResultMapper
     */
    public TestResultDataObject findTestResult(int id) throws
        ClassNotFoundException, SQLException {
        return testResultMapper.find(id);
    }

    /**
     * Passes the CRUB operation to the mapper
     *
     * @param testResult TestResultDataObject to be updated
     * @return a TestResultDataObject containing the data
     * @throws SQLException
     * @throws ClassNotFoundException
     * @see TestResultMapper
     */
    public TestResultDataObject updateTestResult(TestResultDataObject
        testResult) throws SQLException, ClassNotFoundException {
        return testResultMapper.upate(testResult);
    }

    /**
     * Passes the CRUB operation to the mapper
     *
     * @param testResult TestResultDataObject to be removed
     * @throws SQLException
     * @throws ClassNotFoundException
     * @see TestResultMapper
     */
    public void removeTestResult(TestResultDataObject testResult) throws
        SQLException, ClassNotFoundException {
        testResultMapper.delete(testResult);
    }

    /**
```

```
     * Returns the databaseConnection
     *
     * @return SQlConnection to the database
     * @throws ClassNotFoundException
     * @throws SQLException
     * @see Connection
     */
    public Connection getConnection() throws ClassNotFoundException,
        SQLException {
        return connection.getConnection();
    }

    /**
     * Closes the database connection.
     *
     * @throws SQLException
     */
    public void closeConnection() throws SQLException {
        connection.closeConnection();
    }
}
```

Listing A.7: be.hogent.bulksolvingstatistics.ui.BulkSolvingStatisticsGui

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.bulksolvingstatistics.ui;

/**
 * BulkSolvingStatisticsGui.java (UTF-8)
 *
 * This Is the Gui class for the neural network based captcha solving
 * system used for generating the statistical results used in my thesis.
 * It acts as the controller and facade for the UI
 *
 * 2013/05/19
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class BulkSolvingStatisticsGui {

}
```

Listing A.8: be.hogent.captchabuilder.builder.BackgroundParser

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.builder;

import be.hogent.captchabuilder.elementcreator.producer.background.
    BackgroundProducerBuilder;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import be.hogent.captchabuilder.util.enums.producer.BackgroundProducerType;
import java.util.Arrays;
import org.apache.commons.cli.ParseException;

/**
 * BackgroundParser.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/17
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.8
 * @version 1.0.13
 */
public class BackgroundParser {

    public static CaptchaBuilder parse(String[] buildSequenceOptions,
        CaptchaBuilder builder) throws ParseException {
        if (buildSequenceOptions.length == 0) {
            //return builder.addBackground();
            builder.addBuildSequence(new BackgroundProducerBuilder(
                BackgroundProducerType.TRANSPARENT));
            return builder;
        }

        if (buildSequenceOptions.length > 1) {
            throw new ParseException("Background␣takes␣a␣max␣of␣1␣arguments"
                );
        }

        for (String backgroundOption : buildSequenceOptions) {
            if (!backgroundOption.isEmpty()) {
                try {
                    String[] optionArgs = backgroundOption.split(
                        CaptchaConstants.buildSequencelvl3Delim);
                    BackgroundProducerType backgroundProducerType =
                        BackgroundProducerType.valueOf(optionArgs[0]);
                    String[] backgroundOptionArgs = Arrays.copyOfRange(
                        optionArgs, 1, optionArgs.length);
                    return parseBackgroundProducer(backgroundProducerType,
                        backgroundOptionArgs, builder);
                } catch (IllegalArgumentException e) {
                    throw new ParseException(e.getMessage());
                }
            }

        }

        return builder;
    }

    private static CaptchaBuilder parseBackgroundProducer(
        BackgroundProducerType backgroundProducerType, String[]
        backgroundProducerOptions, CaptchaBuilder builder) throws
        ParseException {
        BackgroundProducerBuilder backgroundProducerBuilder = new
            BackgroundProducerBuilder(backgroundProducerType);

        if (backgroundProducerOptions.length == 0) {
            //return builder.addBackground(backgroundProducerBuilder.create
                ());
            builder.addBuildSequence(backgroundProducerBuilder);
            return builder;
        }

        if (backgroundProducerOptions.length > BackgroundProducerOptions.
            values().length) {
            throw new ParseException("BackgroundProducer␣takes␣a␣max␣of␣" +
                BackgroundProducerOptions.values().length + "␣arguments");
        }

        for (String backgroundProducerOption : backgroundProducerOptions) {
            if (!backgroundProducerOption.isEmpty()) {
                try {
                    String[] optionArgs = backgroundProducerOption.split(
                        CaptchaConstants.buildSequencelvl4Delim);
                    BackgroundProducerOptions backgroundProducerOptionType =
                         BackgroundProducerOptions.valueOf(optionArgs[0]);
                    String[] backgroundProducerOptionArgs = Arrays.
                        copyOfRange(optionArgs, 1, optionArgs.length);
                    backgroundProducerBuilder =
                        parseBackgroundProducerOption(
                        backgroundProducerOptionType,
                        backgroundProducerOptionArgs,
                        backgroundProducerBuilder);
                } catch (IllegalArgumentException e) {
                    throw new ParseException(e.getMessage());
                }
            }

        }

        //return builder.addBackground(backgroundProducerBuilder.create());
        builder.addBuildSequence(backgroundProducerBuilder);
        return builder;
    }

    private static BackgroundProducerBuilder parseBackgroundProducerOption(
        BackgroundProducerOptions backgroundProducerOptionType, String[]
        backgroundProducerOptionArgs, BackgroundProducerBuilder
        backgroundProducerBuilder) throws ParseException {
        if (backgroundProducerOptionArgs.length != 1) {
            throw new ParseException("BackgroundProducer␣option␣" +
                backgroundProducerOptionType.name() + "␣only␣takes␣1␣
                argument");
        }

        String[] colorArgs = backgroundProducerOptionArgs[0].split(
            CaptchaConstants.buildSequencelvl5Delim);

        switch (backgroundProducerOptionType) {
            case COLORS1:
                try {
                    return backgroundProducerBuilder.setColorRange1(
                        ColorsParser.parse(colorArgs));
                } catch (NumberFormatException e) {
                    throw new ParseException("Background␣colors1␣has␣invalid
                        ␣formatted␣numbers");
                }
            case COLORS2:
                try {
                    return backgroundProducerBuilder.setColorRange2(
                        ColorsParser.parse(colorArgs));
                } catch (NumberFormatException e) {
                    throw new ParseException("Background␣colors2␣has␣invalid
                        ␣formatted␣numbers");
                }
            default:
                throw new ParseException("BackgroundProducer␣option␣not␣
                    found:␣" + backgroundProducerOptionType.name());
        }
    }

    enum BackgroundProducerOptions {

        COLORS1,
        COLORS2;
    }
}
```

Listing A.9: be.hogent.captchabuilder.builder.BorderParser

```
/*
 * The MIT License
 *
 * Copyright 2013 piva.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.builder;

import be.hogent.captchabuilder.elementcreator.producer.border.
    BorderProducerBuilder;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import be.hogent.captchabuilder.util.enums.producer.BorderProducerType;
import java.util.Arrays;
import org.apache.commons.cli.ParseException;

/**
 * BorderParser.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/17
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.8
 * @version 1.0.13
 */
class BorderParser {

    static CaptchaBuilder parse(String[] borderOptions, CaptchaBuilder
        builder) throws ParseException {
        if (borderOptions.length == 0) {
            //return builder.addBorder();
            builder.addBuildSequence(new BorderProducerBuilder(
                BorderProducerType.SOLID));
            return builder;
        }

        if (borderOptions.length > 1) {
            throw new ParseException("Border␣takes␣a␣max␣of␣1␣arguments");
        }

        for (String borderOption : borderOptions) {
            if (!borderOption.isEmpty()) {
                try {
                    String[] optionArgs = borderOption.split(
                        CaptchaConstants.buildSequencelvl3Delim);
                    BorderProducerType borderProducerType =
                        BorderProducerType.valueOf(optionArgs[0]);
                    String[] borderOptionArgs = Arrays.copyOfRange(
                        optionArgs, 1, optionArgs.length);
                    return parseBorderProducer(borderProducerType,
                        borderOptionArgs, builder);
                } catch (IllegalArgumentException e) {
                    throw new ParseException(e.getMessage());
                }
            }
        }

        return builder;
    }

    private static CaptchaBuilder parseBorderProducer(BorderProducerType
        borderProducerType, String[] borderProducerOptions, CaptchaBuilder
        builder) throws ParseException {
        BorderProducerBuilder borderProducerBuilder = new
            BorderProducerBuilder(borderProducerType);

        if (borderProducerOptions.length == 0) {
            //return builder.addBorder(borderProducerBuilder.create());
            builder.addBuildSequence(borderProducerBuilder);
            return builder;
        }

        if (borderProducerOptions.length > BorderProducerOptions.values().
            length) {
            throw new ParseException("BorderProducer␣takes␣a␣max␣of␣" +
                BorderProducerOptions.values().length + "␣arguments");
        }

        for (String boderproducerOption : borderProducerOptions) {
            if (!boderproducerOption.isEmpty()) {
                try {
                    String[] optionArgs = boderproducerOption.split(
                        CaptchaConstants.buildSequencelvl4Delim);
                    BorderProducerOptions borderProducerOptionType =
                        BorderProducerOptions.valueOf(optionArgs[0]);
                    String[] borderProducerOptionArgs = Arrays.copyOfRange(
                        optionArgs, 1, optionArgs.length);
                    borderProducerBuilder = parseBorderProducerOption(
                        borderProducerOptionType, borderProducerOptionArgs,
                        borderProducerBuilder);
                } catch (IllegalArgumentException e) {
                    throw new ParseException(e.getMessage());
                }
            }

        }

        //return builder.addBorder(borderProducerBuilder.create());
        builder.addBuildSequence(borderProducerBuilder);
        return builder;
    }

    private static BorderProducerBuilder parseBorderProducerOption(
        BorderProducerOptions borderProducerOptionType, String[]
        borderProducerOptionArgs, BorderProducerBuilder
        borderProducerBuilder) throws ParseException {
        if (borderProducerOptionArgs.length != 1) {
            throw new ParseException("BorderProducer option " +
                borderProducerOptionType.name() + " only takes 1 argument");
        }

        switch (borderProducerOptionType) {
            case COLORS:
                try {
                    String[] colorArgs = borderProducerOptionArgs[0].split(
                        CaptchaConstants.buildSequencelvl5Delim);
                    return borderProducerBuilder.setColorRange(ColorsParser.
                        parse(colorArgs));
                } catch (NumberFormatException e) {
                    throw new ParseException("Border colors has invalid
                        formatted numbers");
                }
            case THICKNESS:
                try {
                    return borderProducerBuilder.setThickness(Integer.
                        parseInt(borderProducerOptionArgs[0]));
                } catch (NumberFormatException e) {
                    throw new ParseException("Border thickness argument has
                        an invalid number format");
                }
            default:
                throw new ParseException("BorderProducer option not found: "
                     + borderProducerOptionType.name());

        }
    }

    enum BorderProducerOptions {

        COLORS,
        THICKNESS;
    }
}
```

Listing A.10: be.hogent.captchabuilder.builder.Captcha

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.builder;

import java.io.Serializable;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.Date;
import javax.imageio.ImageIO;

/**
 * Captcha.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/17
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.7
 * @version 1.0.7
 */
public class Captcha implements Serializable {

    private static final long serialVersionUID = 617954136L;
    private String answer;
    private String buildSequence;
    private boolean caseSensative;
    private Date timestamp;
    private BufferedImage captchaImage;
    protected Captcha(String buildSequence, String answer, boolean
        caseSensative, BufferedImage captchaImage, Date timestamp) {
        this.buildSequence = buildSequence;
        this.answer = answer;
        this.captchaImage = captchaImage;
        this.timestamp = timestamp;
        this.caseSensative = caseSensative;
    }

    public boolean isCorrect(String response) {
        if (caseSensative) {
            return answer.equals(response);
        } else {
            return answer.equalsIgnoreCase(response);
        }
    }

    public String getAnswer() {
        return answer;
    }

    public BufferedImage getImage() {
        return captchaImage;
    }

    public Date getTimeStamp() {
        return timestamp;
    }

    @Override
    public String toString() {
        return new StringBuilder()
                .append("[Answer: ")
                .append(answer)
                .append("][Case sensative: ")
                .append(caseSensative)
                .append("][Timestamp: ")
                .append(timestamp)
                .append("][Image: ")
                .append(captchaImage)
                .append("][Build Sequence: ")
                .append(buildSequence)
                .append("]")
                .toString();
    }

    private void writeObject(ObjectOutputStream out) throws IOException {
        out.writeObject(buildSequence);
        out.writeObject(answer);
        out.writeObject(caseSensative);
        out.writeObject(timestamp);
        ImageIO.write(captchaImage, "png", ImageIO.createImageOutputStream(
            out));
    }

    private void readObject(ObjectInputStream in) throws IOException,
        ClassNotFoundException {
        buildSequence = (String) in.readObject();
        answer = (String) in.readObject();
        caseSensative = (Boolean) in.readObject();
        timestamp = (Date) in.readObject();
        captchaImage = ImageIO.read(ImageIO.createImageInputStream(in));
    }
}
```

Listing A.11: be.hogent.captchabuilder.builder.CaptchaBuilder

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.builder;

import be.hogent.captchabuilder.elementcreator.CaptchaElementCreatorBuilder;
import be.hogent.captchabuilder.elementcreator.producer.background.
    BackgroundProducer;
import be.hogent.captchabuilder.elementcreator.producer.background.
    BackgroundProducerBuilder;
import be.hogent.captchabuilder.elementcreator.producer.border.
    BorderProducer;
import be.hogent.captchabuilder.elementcreator.producer.noise.NoiseProducer;
import be.hogent.captchabuilder.elementcreator.producer.text.TextProducer;
import be.hogent.captchabuilder.elementcreator.renderer.gimpy.GimpyRenderer;
import be.hogent.captchabuilder.elementcreator.renderer.text.WordRenderer;
import be.hogent.captchabuilder.util.enums.producer.BackgroundProducerType;
import java.awt.AlphaComposite;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.util.ArrayDeque;
import java.util.Date;
import org.apache.commons.cli.ParseException;

/**
 * CaptchaBuilder.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/17
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.7
 * @version 1.0.13
 */
public class CaptchaBuilder {

    private BufferedImage img;
    private BufferedImage bg;
    private boolean caseSensative;
    private String answer;
    private String buildSequence;
    private ArrayDeque<CaptchaElementCreatorBuilder> builders;

    public CaptchaBuilder(int width, int height, String buildSequence)
        throws ParseException {
        this.builders = new ArrayDeque<>();
        this.setBuildSequence(buildSequence);
        img = new BufferedImage(width, height, BufferedImage.TYPE_INT_ARGB);
        answer = "";
    }

    protected CaptchaBuilder addBackground(BackgroundProducer
        backgroundProducer) {
        bg = backgroundProducer.getBackground(img.getWidth(), img.getHeight
            ());
        return this;
    }

    protected CaptchaBuilder addText(TextProducer textProducer, WordRenderer
         wordRenderer) {
        answer += textProducer.getText();
        wordRenderer.render(answer, img);
        return this;
    }

    protected CaptchaBuilder addNoise(NoiseProducer noiseProducer) {
        noiseProducer.makeNoise(img);
        return this;
    }

    protected CaptchaBuilder gimp(GimpyRenderer gimpyRenderer) {
        gimpyRenderer.gimp(img);
        return this;
    }

    protected CaptchaBuilder addBorder(BorderProducer borderProducer) {
        borderProducer.addBorder(img);
        return this;
    }

    public CaptchaBuilder setImageSize(int width, int height) {
        this.img = new BufferedImage(width, height, BufferedImage.
            TYPE_INT_ARGB);
        return this;
    }

    public final CaptchaBuilder setBuildSequence(String buildSequence)
        throws ParseException {
```

```
        if (!buildSequence.equalsIgnoreCase(this.buildSequence)) {
            this.buildSequence = buildSequence.toUpperCase();

            // If the buildSequence has changed then longParse it
            // Before longparsing, empty the elementbuilderDeque
            this.builders.clear();
            // start parsing
            long startTimeLong = System.nanoTime();
            CaptchaBuildSequenceParser.longParse(this);
            long endTimeLong = System.nanoTime();
            double duration = (double) ((endTimeLong - startTimeLong) / Math
                .pow(10, 9));
            System.out.println("Long␣buildSequence␣parsed␣in␣" + duration +
                "␣seconds");
        }

        return this;
    }

    private Captcha build() {
        return new Captcha(buildSequence, answer, caseSensative,
            flattenImage(), new Date());
    }

    public Captcha buildCaptcha() throws ParseException {
        img = new BufferedImage(img.getWidth(), img.getHeight(),
            BufferedImage.TYPE_INT_ARGB);
        answer = "";
        long startTimeShort = System.nanoTime();
        CaptchaBuildSequenceParser.shortParse(this);
        long endTimeShort = System.nanoTime();
        double duration = (double) ((endTimeShort - startTimeShort) / Math.
            pow(10, 9));
        System.out.println("Short␣buildSequence␣parsed␣in␣" + duration + "␣
            seconds");

        return build();
    }

    public int getWidth() {
        return img.getWidth();
    }

    public int getHeight() {
        return img.getHeight();
    }

    public String getBuildSequence() {
        return buildSequence;
    }

    public final ArrayDeque<CaptchaElementCreatorBuilder> getBuilders() {
        return builders;
    }

    public void addBuildSequence(CaptchaElementCreatorBuilder elementBuilder
        ) {
        builders.offer(elementBuilder);
    }

    private BufferedImage flattenImage() {
        BufferedImage rImage;
        if (bg == null) {
            rImage = new BackgroundProducerBuilder(BackgroundProducerType.
                TRANSPARENT).create().getBackground(img.getWidth(), img.
                getHeight());
        } else {
            rImage = bg;
        }

        // Paint the main image over the background
        Graphics2D g = rImage.createGraphics();
        g.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER,
            1.0f));
        g.drawImage(img, null, null);

        return rImage;
    }
}
```

Listing A.12: be.hogent.captchabuilder.builder.CaptchaBuildSequenceParser

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.builder;

import be.hogent.captchabuilder.elementcreator.CaptchaElementCreatorBuilder;
import be.hogent.captchabuilder.elementcreator.producer.background.
    BackgroundProducer;
import be.hogent.captchabuilder.elementcreator.producer.background.
    BackgroundProducerBuilder;
import be.hogent.captchabuilder.elementcreator.producer.border.
    BorderProducer;
import be.hogent.captchabuilder.elementcreator.producer.border.
    BorderProducerBuilder;
```

```
import be.hogent.captchabuilder.elementcreator.producer.noise.NoiseProducer;
import be.hogent.captchabuilder.elementcreator.producer.noise.
    NoiseProducerBuilder;
import be.hogent.captchabuilder.elementcreator.producer.text.TextProducer;
import be.hogent.captchabuilder.elementcreator.producer.text.
    TextProducerBuilder;
import be.hogent.captchabuilder.elementcreator.renderer.gimpy.GimpyRenderer;
import be.hogent.captchabuilder.elementcreator.renderer.gimpy.
    GimpyRendererBuilder;
import be.hogent.captchabuilder.elementcreator.renderer.text.WordRenderer;
import be.hogent.captchabuilder.elementcreator.renderer.text.
    WordRendererBuilder;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import java.util.ArrayDeque;
import java.util.Arrays;
import org.apache.commons.cli.ParseException;

/**
 * CaptchaBuildSequenceParser.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.3
 * @version 1.0.8
 */
public class CaptchaBuildSequenceParser {

    public static void longParse(CaptchaBuilder builder) throws
        ParseException {

        for (String lvl1Arg : builder.getBuildSequence().split(
            CaptchaConstants.buildSequencelvl1Delim)) {
            if (!lvl1Arg.isEmpty()) {
                try {
                    String[] optionArgs = lvl1Arg.split(CaptchaConstants.
                        buildSequencelvl2Delim);
                    BuildSequenceOptions buildSequenceOptionType =
                        BuildSequenceOptions.valueOf(optionArgs[0]);
                    String[] buildSequenceOptions = Arrays.copyOfRange(
                        optionArgs, 1, optionArgs.length);

                    builder = parseBuildSequenceOption(
                        buildSequenceOptionType, buildSequenceOptions,
                        builder);

                } catch (IllegalArgumentException e) {
                    throw new ParseException(e.getMessage());
                }
            }
        }
    }

    private static CaptchaBuilder parseBuildSequenceOption(
        BuildSequenceOptions option, String[] buildSequenceOptions,
        CaptchaBuilder builder) throws ParseException {
        switch (option) {
            case BACKGROUND:
                return BackgroundParser.parse(buildSequenceOptions, builder)
                    ;
            case BORDER:
                return BorderParser.parse(buildSequenceOptions, builder);
            case GIMP:
                return GimpyParser.parse(buildSequenceOptions, builder);
            case NOISE:
                return NoiseParser.parse(buildSequenceOptions, builder);
            case TEXT:
                return TextParser.parse(buildSequenceOptions, builder);
            default:
                throw new ParseException("argument not found: " + option.
                    name());
        }
    }

    public static void shortParse(CaptchaBuilder builder) throws
        ParseException {
        ArrayDeque<CaptchaElementCreatorBuilder> elementBuilders = builder.
            getBuilders().clone();
        ArrayDeque<BuildSequenceOptions> sequence = new ArrayDeque<>();
        for (String lvl1Arg : builder.getBuildSequence().split(
            CaptchaConstants.buildSequencelvl1Delim)) {
            if (!lvl1Arg.isEmpty()) {
                try {
                    String[] optionArgs = lvl1Arg.split(CaptchaConstants.
                        buildSequencelvl2Delim);
                    sequence.offer(BuildSequenceOptions.valueOf(optionArgs
                        [0]));
                } catch (IllegalArgumentException e) {
                    throw new ParseException(e.getMessage());
                }
            }
        }

        for (BuildSequenceOptions buildSequence : sequence) {
            switch (buildSequence) {
                case BACKGROUND: {
                    CaptchaElementCreatorBuilder elementBuilder =
                        elementBuilders.poll();
                    if (elementBuilder instanceof BackgroundProducerBuilder)
                        {
                        builder.addBackground((BackgroundProducer)
                            elementBuilder.create());
                    } else {
                        throw new ParseException("ShortParse Failed .... How 
                            is that possible?\n"
                                + "Class Mismatch: Got " + elementBuilder.
                                    getClass().getSimpleName()
                                + " and expected " +
                                    BackgroundProducerBuilder.class.
                                    getSimpleName());
                    }
                }
                break;
                case BORDER: {
                    CaptchaElementCreatorBuilder elementBuilder =
                        elementBuilders.poll();
                    if (elementBuilder instanceof BorderProducerBuilder) {
                        builder.addBorder((BorderProducer) elementBuilder.
                            create());
                    } else {
```

```
                throw new ParseException("ShortParse Failed .... How 
                    is that possible?\n"
                        + "Class Mismatch: Got " + elementBuilder.
                            getClass().getSimpleName()
                        + " and expected " + BorderProducerBuilder.
                            class.getSimpleName());
            }
        }
        break;
        case GIMP: {
            CaptchaElementCreatorBuilder elementBuilder =
                elementBuilders.poll();
            if (elementBuilder instanceof GimpyRendererBuilder) {
                builder.gimp((GimpyRenderer) elementBuilder.create()
                    );
            } else {
                throw new ParseException("ShortParse Failed .... How 
                    is that possible?\n"
                        + "Class Mismatch: Got " + elementBuilder.
                            getClass().getSimpleName()
                        + " and expected " + GimpyRendererBuilder.
                            class.getSimpleName());
            }
        }
        break;
        case NOISE: {
            CaptchaElementCreatorBuilder elementBuilder =
                elementBuilders.poll();
            if (elementBuilder instanceof NoiseProducerBuilder) {
                builder.addNoise((NoiseProducer) elementBuilder.
                    create());
            } else {
                throw new ParseException("ShortParse Failed .... How 
                    is that possible?\n"
                        + "Class Mismatch: Got " + elementBuilder.
                            getClass().getSimpleName()
                        + " and expected " + NoiseProducerBuilder.
                            class.getSimpleName());
            }
        }
        break;
        case TEXT: {
            CaptchaElementCreatorBuilder elementBuilder1 =
                elementBuilders.poll();
            CaptchaElementCreatorBuilder elementBuilder2 =
                elementBuilders.poll();
            if (elementBuilder1 instanceof TextProducerBuilder &&
                elementBuilder2 instanceof WordRendererBuilder) {
                builder.addText((TextProducer) elementBuilder1.
                    create(), (WordRenderer) elementBuilder2.create
                    ());
            } else {
                throw new ParseException("ShortParse Failed .... How 
                    is that possible?\n"
                        + "Class Mismatch: Got " + elementBuilder1.
                            getClass().getSimpleName()
                        + " and expected " + TextProducerBuilder.
                            class.getSimpleName()
                        + "\n"
                        + "Class Mismatch: Got " + elementBuilder2.
                            getClass().getSimpleName()
```

```
                                    + " and expected " + WordRendererBuilder.
                                        class.getSimpleName());
                        }
                    }
                }
            }
        }

    enum BuildSequenceOptions {

        BACKGROUND,
        BORDER,
        GIMP,
        NOISE,
        TEXT;
    }
}
```

Listing A.13: be.hogent.captchabuilder.builder.ColorsParser

```
/*
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.builder;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import be.hogent.captchabuilder.util.ImageUtil;
import java.util.ArrayList;
import java.util.Arrays;
import org.apache.commons.cli.ParseException;

/**
 * ColorsParser.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/18
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.8
 * @version 1.1.0
 */
public class ColorsParser {

    public static ColorRangeRGBA parse(String[] colorArgs) throws
        ParseException {

        System.out.println("parsing␣colors␣option:␣" + Arrays.deepToString(
            colorArgs));
        ColorOptions colorOptionType = ColorOptions.valueOf(colorArgs[0]);

        switch (colorOptionType) {
            case RANGE:
                if (colorArgs.length != 3) {
                    throw new ParseException("Colors␣range␣Option␣only␣takes
                        ␣2␣argumenst");
                }
                String startHex = "#" + colorArgs[1].toUpperCase();
                String endHex = "#" + colorArgs[2].toUpperCase();
                return new ColorRangeRGBA(ImageUtil.hexadecimalToRGBa(
                    startHex), ImageUtil.hexadecimalToRGBa(endHex));
            case LIST:
                if (colorArgs.length < 2) {
                    throw new ParseException("Colors␣list␣Option␣takes␣at␣
                        least␣2␣argumenst");
                }
                ArrayList<String> hexList = new ArrayList<>();
                for (int i = 1; i < colorArgs.length; i++) {
                    String colorHex = "#"+colorArgs[i].toUpperCase();
                    hexList.add(colorHex);
                }

                return new ColorRangeRGBA(hexList);
            default:
                throw new ParseException("Colors␣option␣not␣found:␣" +
                    colorOptionType.name());
        }
    }
     enum ColorOptions {

        RANGE,
        LIST;
    }
}
```

Listing A.14: be.hogent.captchabuilder.builder.GimpyParser

```
/*
 * The MIT License
 *
 * Copyright 2013 piva.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.builder;

import be.hogent.captchabuilder.elementcreator.renderer.gimpy.
    GimpyRendererBuilder;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import be.hogent.captchabuilder.util.enums.renderer.GimpyRendererType;
import java.util.Arrays;
import org.apache.commons.cli.ParseException;

/**
 * GimpyParser.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/17
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.8
 * @version 1.0.13
 */
class GimpyParser {

    public static CaptchaBuilder parse(String[] buildSequenceOptions,
        CaptchaBuilder builder) throws ParseException {
        if (buildSequenceOptions.length == 0) {
            //return builder.gimp();
            builder.addBuildSequence(new GimpyRendererBuilder(
                GimpyRendererType.RIPPLE));
            return builder;
        }

        if (buildSequenceOptions.length > GimpyRendererOptions.values().
            length) {
            throw new ParseException("Background takes a max of " +
                GimpyRendererOptions.values().length + " arguments");
        }
        for (String gimpyOption : buildSequenceOptions) {
            if (!gimpyOption.isEmpty()) {
                try {
                    String[] optionArgs = gimpyOption.split(CaptchaConstants
                        .buildSequencelvl3Delim);
                    GimpyRendererType gimpyRenenderType = GimpyRendererType.
                        valueOf(optionArgs[0]);
                    String[] gimpyOptions = Arrays.copyOfRange(optionArgs,
                        1, optionArgs.length);
                    return parseGimpyRenderer(gimpyRenenderType,
                        gimpyOptions, builder);
                } catch (IllegalArgumentException e) {
                    throw new ParseException(e.getMessage());
                }
            }
        }

        return builder;
    }

    private static CaptchaBuilder parseGimpyRenderer(GimpyRendererType
        gimpyRendererType, String[] gimpyOptions, CaptchaBuilder builder)
        throws ParseException {
        GimpyRendererBuilder gimpyRendererBuilder = new GimpyRendererBuilder
            (gimpyRendererType);

        if (gimpyOptions.length == 0) {
            //return builder.gimp(gimpyRendererBuilder.create());
            builder.addBuildSequence(gimpyRendererBuilder);
            return builder;
        }

        if (gimpyOptions.length > GimpyRendererOptions.values().length) {
            throw new ParseException("BackgroundProducer␣takes␣a␣max␣of␣" +
                GimpyRendererOptions.values().length + "␣arguments");
        }

        for (String gimpyRendererOption : gimpyOptions) {
            String[] optionArgs = gimpyRendererOption.split(CaptchaConstants
                .buildSequencelvl4Delim);
            try {
                GimpyRendererOptions gimpyRendererOptionType =
                    GimpyRendererOptions.valueOf(optionArgs[0]);
                String[] gimpyRendererOptionArgs = Arrays.copyOfRange(
                    optionArgs, 1, optionArgs.length);
                gimpyRendererBuilder = parseGimpyRendererOption(
                    gimpyRendererOptionType, gimpyRendererOptionArgs,
                    gimpyRendererBuilder);
            } catch (IllegalArgumentException e) {
                throw new ParseException(e.getMessage());
            }
        }

        //return builder.gimp(gimpyRendererBuilder.create());
        builder.addBuildSequence(gimpyRendererBuilder);
        return builder;
    }

    private static GimpyRendererBuilder parseGimpyRendererOption(
        GimpyRendererOptions gimpyRendererOptionType, String[]
        gimpyRendererOptionArgs, GimpyRendererBuilder gimpyRendererBuilder)
        throws ParseException {
```

```
        if (gimpyRendererOptionArgs.length != 1) {
            throw new ParseException("GimpyRenderer option " +
                gimpyRendererOptionType.name() + " only takes 1 argument");
        }

        String arg = gimpyRendererOptionArgs[0];
        String[] colorArgs;

        switch (gimpyRendererOptionType) {
            case DOUBLE1:
                try {
                    return gimpyRendererBuilder.setD1(Double.parseDouble(arg
                        ));
                } catch (NumberFormatException e) {
                    throw new ParseException("Gimp double1 argument has an
                        invalid number format");
                }
            case DOUBLE2:
                try {
                    return gimpyRendererBuilder.setD2(Double.parseDouble(arg
                        ));
                } catch (NumberFormatException e) {
                    throw new ParseException("Gimp double2 argument has an
                        invalid number format");
                }
            case COLORS1:
                try {
                    colorArgs = arg.split(CaptchaConstants.
                        buildSequencelvl5Delim);
                    return gimpyRendererBuilder.setColorRange1(ColorsParser.
                        parse(colorArgs));
                } catch (NumberFormatException e) {
                    throw new ParseException("Gimp colors1 has invalid
                        formatted numbers");
                }
            case COLORS2:
                try {
                    colorArgs = arg.split(CaptchaConstants.
                        buildSequencelvl5Delim);
                    return gimpyRendererBuilder.setColorRange2(ColorsParser.
                        parse(colorArgs));
                } catch (NumberFormatException e) {
                    throw new ParseException("Border colors2 has invalid
                        formatted numbers");
                }
            default:
                throw new ParseException("GimpyRenderer option not found: "
                    + gimpyRendererOptionType.name());
        }
    }

    enum GimpyOptions {

        DEFAULT;
    }

    enum GimpyRendererOptions {

        DOUBLE1,
        DOUBLE2,
        COLORS1,
        COLORS2;
    }
}
```

Listing A.15: be.hogent.captchabuilder.builder.NoiseParser

```
/*
 * The MIT License
 *
 * Copyright 2013 piva.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.builder;

import be.hogent.captchabuilder.elementcreator.producer.noise.
    NoiseProducerBuilder;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import be.hogent.captchabuilder.util.enums.producer.NoiseProducerType;
import java.util.Arrays;
import org.apache.commons.cli.ParseException;

/**
 * NoiseParser.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/17
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.8
 * @version 1.0.13
 */
class NoiseParser {

    static CaptchaBuilder parse(String[] buildSequenceOptions,
        CaptchaBuilder builder) throws ParseException {
```

```
        if (buildSequenceOptions.length == 0) {
            //return builder.addNoise();
            builder.addBuildSequence(new NoiseProducerBuilder(
                NoiseProducerType.CURVEDLINE));
            return builder;
        }

        if (buildSequenceOptions.length > NoiseOptions.values().length) {
            throw new ParseException("Noise␣takes␣a␣max␣of␣" + NoiseOptions.
                values().length + "␣arguments");
        }

        for (String noiseOption : buildSequenceOptions) {
            if (!noiseOption.isEmpty()) {
                try {
                    String[] optionArgs = noiseOption.split(CaptchaConstants
                        .buildSequencelvl3Delim);
                    NoiseProducerType bgProdBuilder = NoiseProducerType.
                        valueOf(optionArgs[0]);
                    String[] noiseOptions = Arrays.copyOfRange(optionArgs,
                        1, optionArgs.length);
                    return parseNoiseProducer(bgProdBuilder, noiseOptions,
                        builder);
                } catch (IllegalArgumentException e) {
                    throw new ParseException(e.getMessage());
                }
            }
        }

        return builder;
    }

    private static CaptchaBuilder parseNoiseProducer(NoiseProducerType
        noiseProducerType, String[] noiseProducerOptions, CaptchaBuilder
        builder) throws ParseException {
        NoiseProducerBuilder noiseProducerBuilder = new NoiseProducerBuilder
            (noiseProducerType);

        if (noiseProducerOptions.length == 0) {
            //return builder.addNoise(noiseProducerBuilder.create());
            builder.addBuildSequence(noiseProducerBuilder);
            return builder;
        }

        if (noiseProducerOptions.length > NoiseProducerOptions.values().
            length) {
            throw new ParseException("NoiseProducer␣takes␣a␣max␣of␣" +
                NoiseProducerOptions.values().length + "␣arguments");
        }

        for (String noiseProducerOption : noiseProducerOptions) {
            String[] optionArgs = noiseProducerOption.split(CaptchaConstants
                .buildSequencelvl4Delim);
            try {
                NoiseProducerOptions noiseProducerOptionType =
                    NoiseProducerOptions.valueOf(optionArgs[0]);
                String[] noiseProducerOptionArgs = Arrays.copyOfRange(
                    optionArgs, 1, optionArgs.length);
                noiseProducerBuilder = parseNoiseProducerOption(
                    noiseProducerOptionType, noiseProducerOptionArgs,
                    noiseProducerBuilder);
            } catch (IllegalArgumentException e) {
                throw new ParseException(e.getMessage());
            }
        }

        //return builder.addNoise(noiseProducerBuilder.create());
        builder.addBuildSequence(noiseProducerBuilder);
        return builder;
    }

    private static NoiseProducerBuilder parseNoiseProducerOption(
        NoiseProducerOptions noiseProducerOptionType, String[]
        noiseProducerOptionArgs, NoiseProducerBuilder noiseProducerBuilder)
        throws ParseException {
        if (noiseProducerOptionArgs.length != 1) {
            throw new ParseException("NoiseProducer option " +
                noiseProducerOptionType.name() + " only takes 1 argument");
        }

        switch (noiseProducerOptionType) {
            case COLORS:
                try {
                    return noiseProducerBuilder.setColorRange(ColorsParser.
                        parse(noiseProducerOptionArgs[0].split(
                        CaptchaConstants.buildSequencelvl5Delim)));
                } catch (NumberFormatException e) {
                    throw new ParseException("Noise colors has invalid
                        formatted numbers");
                }
            case THICKNESS:
                try {
                    return noiseProducerBuilder.setThickness(Float.
                        parseFloat(noiseProducerOptionArgs[0]));
                } catch (NumberFormatException e) {
                    throw new ParseException("Noise thickness argument has
                        an invalid number format");
                }
            default:
                throw new ParseException("NoiseProducer option not found: "
                    + noiseProducerOptionType.name());
        }
    }

    enum NoiseOptions {

        DEFAULT;
    }

    enum NoiseProducerOptions {

        COLORS,
        THICKNESS;
    }
}
```

Listing A.16: be.hogent.captchabuilder.builder.TextParser

```
/*
 * The MIT License
 *
 * Copyright 2013 piva.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.builder;

import be.hogent.captchabuilder.elementcreator.producer.text.
    TextProducerBuilder;
import be.hogent.captchabuilder.elementcreator.renderer.text.
    WordRendererBuilder;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import be.hogent.captchabuilder.util.enums.producer.TextProducerType;
import be.hogent.captchabuilder.util.enums.renderer.WordRendererType;
import java.awt.Font;
import java.util.ArrayList;
import java.util.Arrays;
import org.apache.commons.cli.ParseException;

/**
 * TextParser.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/18
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.8
 * @version 1.0.13
 */
class TextParser {

    private static TextProducerBuilder textProducerBuilder = new
        TextProducerBuilder(TextProducerType.REDUCED_ALPHANUMERIC);
    private static WordRendererBuilder wordRendererBuilder = new
        WordRendererBuilder(WordRendererType.DEFAULT);

    static CaptchaBuilder parse(String[] buildSequenceOptions,
        CaptchaBuilder builder) throws ParseException {
```

```
        for (String textOptionArg : buildSequenceOptions) {
            if (!textOptionArg.isEmpty()) {
                try {
                    String[] optionArgs = textOptionArg.split(
                        CaptchaConstants.buildSequencelvl3Delim);
                    TextOptions textOptionType = TextOptions.valueOf(
                        optionArgs[0]);
                    String[] textOptions = Arrays.copyOfRange(optionArgs, 1,
                         optionArgs.length);

                    parseTextOption(textOptionType, textOptions, builder);
                } catch (IllegalArgumentException e) {
                    throw new ParseException(e.getMessage());
                }

            }
        }

        //return builder.addText(textProducerBuilder.create(),
            wordRendererBuilder.create());
        builder.addBuildSequence(textProducerBuilder);
        builder.addBuildSequence(wordRendererBuilder);
        return builder;
    }

    private static void parseTextOption(TextOptions textOptionType, String[]
         textOptions, CaptchaBuilder builder) throws ParseException {

        switch (textOptionType) {
            case TEXTPRODUCER:
                textProducerBuilder = TextProducerParser.parse(textOptions,
                    textProducerBuilder);
                break;
            case WORDRENDERER:
                wordRendererBuilder = WordRendererParser.parse(textOptions,
                    wordRendererBuilder);
                break;
            default:
                throw new ParseException("Text argument not found: " +
                    textOptionType.name());
        }
    }

    private static class TextProducerParser {

        private static TextProducerBuilder parse(String[]
            textProducerOptions, TextProducerBuilder builder) throws
            ParseException {
            if (textProducerOptions.length == 0) {
                builder = new TextProducerBuilder(TextProducerType.
                    REDUCED_ALPHANUMERIC);
            }

            if (textProducerOptions.length > 1) {
                throw new ParseException("TextProducer takes a max of 1
                    argument");
            }

            for (String textProducerOption : textProducerOptions) {
                if (!textProducerOption.isEmpty()) {
                    String[] optionArgs = textProducerOption.split(
                        CaptchaConstants.buildSequencelvl4Delim);
                TextProducerType textProducerType = TextProducerType.
                    valueOf(optionArgs[0]);
                String[] textProducerOptionArgs = Arrays.copyOfRange(
                    optionArgs, 1, optionArgs.length);

                builder = new TextProducerBuilder(textProducerType);
                builder = parseTextProducerOption(textProducerType,
                    textProducerOptionArgs, builder);
            }
        }

        return builder;
    }

    private static TextProducerBuilder parseTextProducerOption(
        TextProducerType textProducerType, String[]
        textProducerOptionArgs, TextProducerBuilder builder) throws
        ParseException {
        if (textProducerOptionArgs.length == 0) {
            builder = new TextProducerBuilder(textProducerType);
        }

        if (textProducerOptionArgs.length > TextProducerOptions.values()
            .length) {
            throw new ParseException("TextProducerType␣takes␣a␣max␣of␣"
                + TextProducerOptions.values().length + "␣arguments");
        }

        for (String textProducerTypeOption : textProducerOptionArgs) {
            if (!textProducerTypeOption.isEmpty()) {
                String[] optionArgs = textProducerTypeOption.split(
                    CaptchaConstants.buildSequencelvl5Delim);
                TextProducerOptions textProducerOptionType =
                    TextProducerOptions.valueOf(optionArgs[0]);
                String[] textProducerTypeOptionArgs = Arrays.copyOfRange
                    (optionArgs, 1, optionArgs.length);

                builder = parseTextProducerTypeOption(
                    textProducerOptionType, textProducerTypeOptionArgs,
                    builder);
            }
        }

        return builder;
    }

    private static TextProducerBuilder parseTextProducerTypeOption(
        TextProducerOptions textProducerOptionType, String[]
        textProducerTypeOptionArgs, TextProducerBuilder builder) throws
        ParseException {
        if (textProducerTypeOptionArgs.length != 1) {
            throw new ParseException("TextProducerOption␣" +
                textProducerOptionType.name() + "␣only␣takes␣one␣
                argument");
        }

        switch (textProducerOptionType) {
            case MINLENGTH:
                try {
                    return builder.setMinLenght(Integer.parseInt(
                        textProducerTypeOptionArgs[0]));
                } catch (NumberFormatException e) {
                    throw new ParseException("Text␣TextProducer␣
                        MinLength␣argument␣has␣an␣invalid␣number␣format"
                        );
                }
            case MAXLENGTH:
                try {
                    return builder.setMaxLenght(Integer.parseInt(
                        textProducerTypeOptionArgs[0]));
                } catch (NumberFormatException e) {
                    throw new ParseException("Text␣TextProducer␣
                        MaxLength␣argument␣has␣an␣invalid␣number␣format"
                        );
                }
            default:
                throw new ParseException("TextProducerOptionType␣not␣
                    found:␣" + textProducerOptionType.name());
        }
    }
}

private static class WordRendererParser {

    private static WordRendererBuilder parse(String[]
        wordRendererOptions, WordRendererBuilder builder) throws
        ParseException {
        if (wordRendererOptions.length == 0) {
            builder = new WordRendererBuilder(WordRendererType.DEFAULT);
        }

        if (wordRendererOptions.length > 1) {
            throw new ParseException("WordRenderer␣takes␣a␣max␣of␣1␣
                argument");
        }

        for (String wordRendererOption : wordRendererOptions) {
            if (!wordRendererOption.isEmpty()) {
                String[] optionArgs = wordRendererOption.split(
                    CaptchaConstants.buildSequencelvl4Delim);
                WordRendererType wordRendererType = WordRendererType.
                    valueOf(optionArgs[0]);
                String[] wordRendererOptionArgs = Arrays.copyOfRange(
                    optionArgs, 1, optionArgs.length);

                builder = parseWordRendererOption(wordRendererType,
                    wordRendererOptionArgs, builder);
            }
        }

        return builder;
    }

    private static WordRendererBuilder parseWordRendererOption(
        WordRendererType wordRendererType, String[]
        wordRendererOptionArgs, WordRendererBuilder builder) throws
        ParseException {
        if (wordRendererOptionArgs.length == 0) {
            return builder;
        }

        if (wordRendererOptionArgs.length > WordRendererOptions.values()
            .length) {
```

```
            throw new ParseException("WordRendererType takes a max of "
                + WordRendererOptions.values().length + " arguments");
        }

        for (String wordRendererTypeOption : wordRendererOptionArgs) {
            if (!wordRendererTypeOption.isEmpty()) {
                String[] optionArgs = wordRendererTypeOption.split(
                    CaptchaConstants.buildSequencelvl5Delim);
                WordRendererOptions wordRendererOptionType =
                    WordRendererOptions.valueOf(optionArgs[0]);
                String[] wordRendererTypeOptionArgs = Arrays.copyOfRange
                    (optionArgs, 1, optionArgs.length);

                builder = parseWordRendererTypeOption(
                    wordRendererOptionType, wordRendererTypeOptionArgs,
                    builder);
            }
        }

        return builder;
    }

    private static WordRendererBuilder parseWordRendererTypeOption(
        WordRendererOptions wordRendererOptionType, String[]
        wordRendererTypeOptionArgs, WordRendererBuilder builder) throws
        ParseException {
        switch (wordRendererOptionType) {
            case COLORS:
                try {
                    if (wordRendererTypeOptionArgs.length != 1) {
                        throw new ParseException("WordRendererOption " +
                             wordRendererOptionType.name() + " only
                            takes one argument");
                    }
                    String[] colorArgs = wordRendererTypeOptionArgs[0].
                        split(CaptchaConstants.buildSequencelvl6Delim);
                    return builder.setColorRange(ColorsParser.parse(
                        colorArgs));
                } catch (NumberFormatException e) {
                    throw new ParseException("Text WordRenderer colors
                        has invalid formatted numbers");
                }
            case FONTS:
                if (wordRendererTypeOptionArgs.length < 1) {
                    throw new ParseException("WordRendererOption " +
                        wordRendererOptionType.name() + " only takes one
                         argument");
                }
                ArrayList<Font> fonts = new ArrayList<>();
                for (String fontString : wordRendererTypeOptionArgs) {
                    String[] fontArgs = fontString.split(
                        CaptchaConstants.buildSequencelvl6Delim);
                    fonts.add(new Font(fontArgs[0], Integer.parseInt(
                        fontArgs[1]), Integer.parseInt(fontArgs[2])));
                }
                return builder.setFonts(fonts);
            case STROKE:
                if (wordRendererTypeOptionArgs.length != 1) {
                    throw new ParseException("WordRendererOption " +
                        wordRendererOptionType.name() + " only takes one
                         argument");
                }
                try {
                    return builder.setStrokeWidth(Float.parseFloat(
                        wordRendererTypeOptionArgs[0]));
                } catch (NumberFormatException e) {
                    throw new ParseException("Text␣WordRenderer␣Stroke␣
                        argument␣has␣an␣invalid␣number␣format");
                }
            case XOFF:
                if (wordRendererTypeOptionArgs.length != 1) {
                    throw new ParseException("WordRendererOption␣" +
                        wordRendererOptionType.name() + "␣only␣takes␣one
                        ␣argument");
                }
                try {
                    return builder.setXOffset(Double.parseDouble(
                        wordRendererTypeOptionArgs[0]));
                } catch (NumberFormatException e) {
                    throw new ParseException("Text␣WordRenderer␣XOFF␣
                        argument␣has␣an␣invalid␣number␣format");
                }
            case YOFF:
                if (wordRendererTypeOptionArgs.length != 1) {
                    throw new ParseException("WordRendererOption␣" +
                        wordRendererOptionType.name() + "␣only␣takes␣one
                        ␣argument");
                }
                try {
                    return builder.setYOffset(Double.parseDouble(
                        wordRendererTypeOptionArgs[0]));
                } catch (NumberFormatException e) {
                    throw new ParseException("Text␣WordRenderer␣YOFF␣
                        argument␣has␣an␣invalid␣number␣format");
                }
            default:
                throw new ParseException("WordRendeereOptionType␣not␣
                    found:␣" + wordRendererOptionType.name());
        }
    }
}

enum TextOptions {

    TEXTPRODUCER,
    WORDRENDERER;
}

enum TextProducerOptions {

    MINLENGTH,
    MAXLENGTH;
}

enum WordRendererOptions {

    COLORS,
    FONTS,
    STROKE,
    XOFF,
    YOFF;
}

enum FontOptions {
```

```
            FONTNAME,
            FONTSTYLE,
            FONTSIZE;
    }
}
```

Listing A.17: be.hogent.captchabuilder.elementcreator.CaptchaElementCreatorBuilder

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator;

/**
 * CaptchaElementCreatorBuilder.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/18
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.15
 * @version 1.1.0
 */
public interface CaptchaElementCreatorBuilder<T> {

    public T create();
}
```

A.19 Package be.hogent.captchabuilder.elementcreator.pro

A.20 Package be.hogent.captchabuilder.elementcreator.ren

Listing A.18: be.hogent.captchabuilder.util.ArrayUtil

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.util;

import java.util.Arrays;

/**
 * ArrayUtil.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/15
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.2
 * @version 1.0.2
 */
public abstract class ArrayUtil<T> {

    @SuppressWarnings("unchecked")
    public static <T> T[] concat(T[] first, T[]... rest) {
        int totalLength = first.length;
        for (T[] array : rest) {
            totalLength += array.length;
        }
        T[] result = Arrays.copyOf(first, totalLength);
        int offset = first.length;
        for (T[] array : rest) {
            System.arraycopy(array, 0, result, offset, array.length);
            offset += array.length;
        }
        return result;
    }

    public static char[] concat(char[] first, char[]... rest) {
        int totalLength = first.length;
        for (char[] array : rest) {
            totalLength += array.length;
        }
        char[] result = Arrays.copyOf(first, totalLength);
        int offset = first.length;
        for (char[] array : rest) {
            System.arraycopy(array, 0, result, offset, array.length);
            offset += array.length;
        }
        return result;
    }

    public static int[] concat(int[] first, int[]... rest) {
        int totalLength = first.length;
        for (int[] array : rest) {
            totalLength += array.length;
        }
        int[] result = Arrays.copyOf(first, totalLength);
        int offset = first.length;
        for (int[] array : rest) {
            System.arraycopy(array, 0, result, offset, array.length);
            offset += array.length;
        }
        return result;
    }

    public static double[] concat(double[] first, double[]... rest) {
        int totalLength = first.length;
        for (double[] array : rest) {
            totalLength += array.length;
        }
        double[] result = Arrays.copyOf(first, totalLength);
        int offset = first.length;
        for (double[] array : rest) {
            System.arraycopy(array, 0, result, offset, array.length);
            offset += array.length;
        }
        return result;
    }

    public static float[] concat(float[] first, float[]... rest) {
        int totalLength = first.length;
        for (float[] array : rest) {
            totalLength += array.length;
        }
        float[] result = Arrays.copyOf(first, totalLength);
        int offset = first.length;
        for (float[] array : rest) {
            System.arraycopy(array, 0, result, offset, array.length);
            offset += array.length;
        }
        return result;
    }

    public static byte[] concat(byte[] first, byte[]... rest) {
        int totalLength = first.length;
        for (byte[] array : rest) {
            totalLength += array.length;
        }
        byte[] result = Arrays.copyOf(first, totalLength);
        int offset = first.length;
        for (byte[] array : rest) {
            System.arraycopy(array, 0, result, offset, array.length);
            offset += array.length;
        }
        return result;
    }

    public static short[] concat(short[] first, short[]... rest) {
        int totalLength = first.length;
        for (short[] array : rest) {
            totalLength += array.length;
        }
        short[] result = Arrays.copyOf(first, totalLength);
        int offset = first.length;
        for (short[] array : rest) {
            System.arraycopy(array, 0, result, offset, array.length);
            offset += array.length;
        }
        return result;
    }

    public static long[] concat(long[] first, long[]... rest) {
        int totalLength = first.length;
        for (long[] array : rest) {
            totalLength += array.length;
        }
        long[] result = Arrays.copyOf(first, totalLength);
        int offset = first.length;
        for (long[] array : rest) {
            System.arraycopy(array, 0, result, offset, array.length);
            offset += array.length;
        }
        return result;
    }

    public static boolean[] concat(boolean[] first, boolean[]... rest) {
        int totalLength = first.length;
        for (boolean[] array : rest) {
            totalLength += array.length;
        }
        boolean[] result = Arrays.copyOf(first, totalLength);
        int offset = first.length;
        for (boolean[] array : rest) {
            System.arraycopy(array, 0, result, offset, array.length);
            offset += array.length;
        }
        return result;
    }
}
```

Listing A.19: be.hogent.captchabuilder.util.CaptchaDAO

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.util;

import java.awt.image.BufferedImage;

/**
 * CaptchaDAO.java (UTF-8)
 *
 * A data access object were all data is read only, used to pass the captcha
 * info to a GUI
 *
 * 2013/04/15
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.2.0
 * @version 1.2.0
 */
public class CaptchaDAO {
    private final BufferedImage image;
    private final String answer;
    private final String parserMessage;

    public CaptchaDAO(BufferedImage image, String answer, String
        parserMessage) {
        this.image = image;
        this.answer = answer;
        this.parserMessage = parserMessage;
    }

    public BufferedImage getImage() {
        return image;
    }

    public String getAnswer() {
        return answer;
    }

    public String getParserMessage() {
        return parserMessage;
    }
}
```

Listing A.20: be.hogent.captchabuilder.util.ColorRangeRGBA

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.util;

import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import java.awt.Color;
import java.util.List;
import java.util.Random;

/**
 * ColorRangeRGBA.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/19
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.1.0
 * @version 1.1.0
 */
public class ColorRangeRGBA {

    private final int startR;
    private final int endR;
    private final int startG;
    private final int endG;
    private final int startB;
    private final int endB;
    private final int startA;
    private final int endA;
    private Random random;
    private boolean listMode;
    private List<String> hexList;

    public ColorRangeRGBA(int MSa) {
        this(MSa, MSa);
    }

    public ColorRangeRGBA(List<String> hexList) {
        this(0);
        this.listMode = true;
        this.hexList = hexList;
    }

    public ColorRangeRGBA(int[] rgba) {
        this(rgba, rgba);
    }

    public ColorRangeRGBA(int r, int g, int b) {
        this(r, g, b, 0);
    }

    public ColorRangeRGBA(int r, int b, int g, int a) {
        this(r, r, g, g, g, g, a, a);
    }

    public ColorRangeRGBA(int[] startRGBa, int[] endRGBa) {
        this(startRGBa[0], endRGBa[0], startRGBa[1], endRGBa[1], startRGBa
            [2], endRGBa[2], startRGBa[3], endRGBa[3]);
    }

    public ColorRangeRGBA(int startMSa, int endMSa) {
        this(ImageUtil.msAccesToRGBa(startMSa), ImageUtil.msAccesToRGBa(
            endMSa));
    }

    public ColorRangeRGBA(int startR, int endR, int startG, int endG, int
        startB, int endB, int startA, int endA) {
        this.random = CaptchaConstants.RANDOM;
        this.startR = startR;
        this.endR = endR;
        this.startG = startG;
        this.endG = endG;
        this.startB = startB;
        this.endB = endB;
        this.startA = startA;
        this.endA = endA;
        this.listMode = false;
    }

    public Color getRandomColorInRange() {
        return new Color(getRandomInRangeR(), getRandomInRangeG(),
            getRandomInRangeB(), getRandomInRangeA());
    }

    public int getRandomMSaccesInRange() {
        return ImageUtil.rgbToMsAcces(getRandomInRangeR(), getRandomInRangeG
            (), getRandomInRangeB());
    }

    public int getRandomInRangeR() {
        if (listMode) {
            return ImageUtil.hexadecimalToRGBa(hexList.get(random.nextInt(
                hexList.size())))[0];
        } else {
            return random8bitNumber(startR, endR);
        }
    }

    public int getRandomInRangeG() {
        if (listMode) {
            return ImageUtil.hexadecimalToRGBa(hexList.get(random.nextInt(
                hexList.size())))[1];
        } else {
            return random8bitNumber(startG, endG);
        }
    }

    public int getRandomInRangeB() {
        if (listMode) {
            return ImageUtil.hexadecimalToRGBa(hexList.get(random.nextInt(
                hexList.size())))[2];
        } else {
            return random8bitNumber(startB, endB);
        }
    }

    public int getRandomInRangeA() {
        if (listMode) {
            return ImageUtil.hexadecimalToRGBa(hexList.get(random.nextInt(
                hexList.size())))[3];
        } else {
            return random8bitNumber(startA, endA);
        }
    }

    private int random8bitNumber(int start, int end) {
        if (start > end) {
            if (random.nextBoolean()) {
                return random8bitNumber(0, end);
            } else {
                return random8bitNumber(start, 256);
            }
        }
        if (start == end) {
            return start;
        } else {
            return random.nextInt(end - start) + start;
        }
    }
```

```
}
```

A.21 Package be.hogent.captchabuilder.util.enums

Listing A.21: be.hogent.captchabuilder.util.ImageUtil

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.util;

import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.Toolkit;
import java.awt.image.BufferedImage;
import java.awt.image.FilteredImageSource;
import java.awt.image.ImageFilter;

/**
 * ImageUtil.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/15
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.6
 * @version 1.0.8
 */
public class ImageUtil {

    public static final void applyFilter(BufferedImage img, ImageFilter
        filter) {
        FilteredImageSource src = new FilteredImageSource(img.getSource(),
            filter);
        Image fImg = Toolkit.getDefaultToolkit().createImage(src);
        Graphics2D g = img.createGraphics();
        g.drawImage(fImg, 0, 0, null, null);
        g.dispose();
    }

    public static final int rgbaToMsAcces(int r, int g, int b, int a) {
         Color c = new Color(r, g, b, a);
        return c.getRGB();
    }

    public static final int rgbToMsAcces(int r, int g, int b) {
        return rgbaToMsAcces(r, g, b, 0);
    }

    public static final int[] msAccesToRGBa(int code) {
        Color c = new Color(code);
        return colorToRGBa(c);
    }

    public static int[] hexadecimalToRGBa(String hex) {
        Color c = Color.decode(hex);
        return colorToRGBa(c);
    }

    private static int[] colorToRGBa(Color c) {
        int[] rgba = new int[4];

        rgba[0] = c.getRed();
        rgba[1] = c.getGreen();
        rgba[2] = c.getBlue();
        rgba[3] = c.getAlpha();

        return rgba;
    }
}
```

Listing A.22: be.hogent.captchacleanup.utils.ImageToArray

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package be.hogent.captchacleanup.utils;

import java.awt.image.BufferedImage;

/**
 *
 * @author Pieter
 */
public class ImageToArray {
    public static boolean[][] colorRangeToBooleanArray(BufferedImage image,
        int startRange, int endRange) {
        boolean[][] array = new boolean[image.getWidth()][image.getHeight()
            ];
        int startR = (startRange >> 16) & 0x000000FF;
        int startG = (startRange >> 8) & 0x000000FF;
        int startB = (startRange) & 0x000000FF;
        int endR = (endRange >> 16) & 0x000000FF;
        int endG = (endRange >> 8) & 0x000000FF;
        int endB = (endRange) & 0x000000FF;

        for (int y = 0; y < image.getHeight(); y++) {
            for (int x = 0; x < image.getWidth(); x++) {
                int RGB = image.getRGB(x, y);
                int alpha = (RGB >> 24) & 0x000000FF;
                boolean inRange = false;
                if (alpha != 0) {
                    int R = (startRange >> 16) & 0x000000FF;
                    int G = (startRange >> 8) & 0x000000FF;
                    int B = (startRange) & 0x000000FF;
                    if (startR <= R && R <= endR && startG <= G && G <= endG
                         && startB <= B && B <= endB) {
                        inRange = true;
                    }
                }
                array[x][y] = inRange;
            }

        }

//          // preview array
//          StringBuilder output;
//          for (int y = 0; y < image.getHeight(); y++) {
//              output = new StringBuilder();
//              for (int x = 0; x < image.getWidth(); x++) {
//                  if (array[x][y]) {
//                      output.append("#");
//                  } else {
//                      output.append(" ");
//                  }
//              }
//              System.out.println(output.toString());
//          }

        // return array
        return array;

    }

    public static double[][] colorRangeToDoubleArray(BufferedImage image,
        int startRange, int endRange) {
        double[][] array = new double[image.getWidth()][image.getHeight()];
        int startR = (startRange >> 16) & 0x000000FF;
        int startG = (startRange >> 8) & 0x000000FF;
        int startB = (startRange) & 0x000000FF;
        int endR = (endRange >> 16) & 0x000000FF;
        int endG = (endRange >> 8) & 0x000000FF;
        int endB = (endRange) & 0x000000FF;

        for (int y = 0; y < image.getHeight(); y++) {
            for (int x = 0; x < image.getWidth(); x++) {
                int RGB = image.getRGB(x, y);
                int alpha = (RGB >> 24) & 0x000000FF;
                if (alpha != 0) {
                    int R = (startRange >> 16) & 0x000000FF;
                    int G = (startRange >> 8) & 0x000000FF;
                    int B = (startRange) & 0x000000FF;
                    if (startR <= R && R <= endR && startG <= G && G <= endG
                         && startB <= B && B <= endB) {
                        array[x][y] = 1;
                    } else {
                        array[x][y] = 0;
                    }
                }
            }

        }

//          // preview array
//          StringBuilder output;
//          for (int y = 0; y < image.getHeight(); y++) {
//              output = new StringBuilder();
//              for (int x = 0; x < image.getWidth(); x++) {
//                  if (array[x][y]>=1) {
//                      output.append("#");
//                  } else {
//                      output.append(" ");
//                  }
//              }
//              System.out.println(output.toString());
//          }

        // return array
        return array;
    }
}
```

Listing A.23: be.hogent.captchacleanup.utils.ImageUtils

```
/*
 * The MIT License
 *
 * Copyright 2013 piva.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchacleanup.utils;

import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.Toolkit;
import java.awt.image.BufferedImage;
import java.awt.image.FilteredImageSource;
import java.awt.image.ImageFilter;
import java.awt.image.ImageProducer;
import java.awt.image.RGBImageFilter;

/**
 * DomainFacade.java (UTF-8)
 *
 * This class will be used a container for static access methods
     manipulating images
 *
 * 2013/04/23
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class ImageUtils {

    public static BufferedImage setColorTransparent(BufferedImage bufImage,
        String cString) {
        Color c = Color.decode(cString);
        return setColorRangeTransparent(bufImage, c, c);
    }

    public static BufferedImage setColorTransparent(BufferedImage bufImage,
        int cInt) {
        Color c= new Color(cInt);
        return setColorRangeTransparent(bufImage, c, c);
    }

    public static BufferedImage setColorRangeTransparent(BufferedImage
        bufImage, String c1, String c2) {
        return setColorRangeTransparent(bufImage, Color.decode(c1), Color.
            decode(c2));
    }

    public static BufferedImage setColorRangeTransparent(BufferedImage
        bufImage, int c1, int c2) {
        return setColorRangeTransparent(bufImage, new Color(c1), new Color(
            c2));
    }

    public static BufferedImage setColorRangeTransparent(BufferedImage
        bufImage, Color c1, Color c2) {
        // Primitive test, just an example
        final int r1 = c1.getRed();
        final int g1 = c1.getGreen();
        final int b1 = c1.getBlue();
        final int r2 = c2.getRed();
        final int g2 = c2.getGreen();
        final int b2 = c2.getBlue();
        ImageFilter filter = new RGBImageFilter() {
            @Override
            public final int filterRGB(int x, int y, int rgb) {

//                int r = (rgb & 0xFF0000) >> 16;
//                int g = (rgb & 0xFF00) >> 8;
//                int b = rgb & 0xFF;

                Color c = new Color(rgb);
                int r = c.getRed();
                int g = c.getGreen();
                int b = c.getBlue();
                if (r >= r1 && r <= r2 && g >= g1 && g <= g2 && b >= b1 && b
                     <= b2) {
                    // Set fully transparent but keep color
                    return rgb & 0xFFFFFF;
                }
                return rgb;
            }
        };

        ImageProducer ip = new FilteredImageSource(bufImage.getSource(),
            filter);
        Image image = Toolkit.getDefaultToolkit().createImage(ip);

        bufImage = new BufferedImage(bufImage.getWidth(), bufImage.getHeight
            (), BufferedImage.TYPE_INT_ARGB);
        Graphics2D g = bufImage.createGraphics();
        g.drawImage(image, 0, 0, null);
        g.dispose();
        return bufImage;
    }
}
```

A.22 Package be.hogent.captchacleanup.utils.textfromimag

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package be.hogent.captchasolvingnetwork.encog_2;

import org.encog.ml.data.specific.BiPolarNeuralData;
import org.encog.neural.thermal.HopfieldNetwork;

/**
 *
 * @author Pieter
 */
public class EncogHopfieldNetworkExample {

    final static int HEIGHT = 10;
    final static int WIDTH = 10;

    public EncogHopfieldNetworkExample() {
    }
    /**
     * The neural network will learn these patterns.
     */
    public static final String[][] PATTERN = {{
            "O␣O␣O␣O␣O␣",
            "␣O␣O␣O␣O␣O",
            "O␣O␣O␣O␣O␣",
            "␣O␣O␣O␣O␣O",
            "O␣O␣O␣O␣O␣",
            "␣O␣O␣O␣O␣O",
            "O␣O␣O␣O␣O␣",
            "␣O␣O␣O␣O␣O",
            "O␣O␣O␣O␣O␣",
            "␣O␣O␣O␣O␣O"},
        {"OO␣␣OO␣␣OO",
            "OO␣␣OO␣␣OO",
            "␣␣OO␣␣OO␣␣",
            "␣␣OO␣␣OO␣␣",
            "OO␣␣OO␣␣OO",
            "OO␣␣OO␣␣OO",
            "␣␣OO␣␣OO␣␣",
            "␣␣OO␣␣OO␣␣",
            "OO␣␣OO␣␣OO",
            "OO␣␣OO␣␣OO"},
        {"OOOOO␣␣␣␣␣",
            "OOOOO␣␣␣␣␣",
            "OOOOO␣␣␣␣␣",
            "OOOOO␣␣␣␣␣",
            "OOOOO␣␣␣␣␣",
            "␣␣␣␣␣OOOOO",
            "␣␣␣␣␣OOOOO",
            "␣␣␣␣␣OOOOO",
            "␣␣␣␣␣OOOOO",
            "␣␣␣␣␣OOOOO"},
        {"O␣␣O␣␣O␣␣O",
            "␣O␣␣O␣␣O␣␣",
            "␣␣O␣␣O␣␣O␣",
            "O␣␣O␣␣O␣␣O",
            "␣O␣␣O␣␣O␣␣",
            "␣␣O␣␣O␣␣O␣",
            "O␣␣O␣␣O␣␣O",
            "␣O␣␣O␣␣O␣␣",
            "␣␣O␣␣O␣␣O␣",
            "O␣␣O␣␣O␣␣O"},
        {"OOOOOOOOOO",
            "O␣␣␣␣␣␣␣␣O",
            "O␣OOOOOO␣O",
            "O␣O␣␣␣␣O␣O",
            "O␣O␣OO␣O␣O",
            "O␣O␣OO␣O␣O",
            "O␣O␣␣␣␣O␣O",
            "O␣OOOOOO␣O",
            "O␣␣␣␣␣␣␣␣O",
            "OOOOOOOOOO"},
        {"␣␣␣␣␣␣␣␣␣␣",
            "␣OOOOOOOO␣",
            "␣OO␣␣␣␣OO␣",
            "␣OO␣␣␣␣OO␣",
            "␣OOOOOOOO␣",
            "␣OOOOOOOO␣",
            "␣OO␣␣␣␣OO␣",
            "␣OO␣␣␣␣OO␣",
            "␣OOOOOOOO␣",
            "␣␣␣␣␣␣␣␣␣␣"},
        {"␣␣␣␣␣␣␣␣␣␣",
            "␣OOOOOOOO␣",
            "␣OO␣␣␣␣OO␣",
            "␣OO␣␣␣␣OO␣",
            "␣OOOOOOOO␣",
            "␣OOOOOOOO␣",
            "␣OO␣␣␣␣OO␣",
            "␣OO␣␣␣␣OO␣",
            "␣OO␣␣␣␣OO␣",
            "␣␣␣␣␣␣␣␣␣␣"}};
    /**
     * The neural network will be tested on these patterns, to see which of
         the
     * last set they are the closest to.
     */
    public static final String[][] PATTERN2 = {{
            "␣␣␣␣␣␣␣␣␣␣",
            "␣␣␣␣␣␣␣␣␣␣",
            "␣␣␣␣␣␣␣␣␣␣",
            "␣␣␣␣␣␣␣␣␣␣",
            "␣␣␣␣␣␣␣␣␣␣",
            "␣O␣O␣O␣O␣O",
            "O␣O␣O␣O␣O␣",
            "␣O␣O␣O␣O␣O",
            "O␣O␣O␣O␣O␣",
            "␣O␣O␣O␣O␣O"},
        {"OOO␣O␣␣␣␣O",
            "␣O␣␣OOO␣OO",
            "␣␣O␣O␣OO␣O",
            "␣OOO␣␣␣O␣␣",
            "OO␣␣O␣␣OOO",
            "␣O␣OOO␣␣␣O",
            "O␣OO␣␣O␣␣O",
            "␣␣␣O␣OOO␣␣",
            "OO␣OOO␣␣O␣",
            "␣O␣␣O␣␣OOO"},
        {"OOOOO␣␣␣␣␣",
            "O␣␣␣O␣OOO␣",
            "O␣␣␣O␣OOO␣",
            "O␣␣␣O␣OOO␣",
            "OOOOO␣␣␣␣␣",
            "␣␣␣␣␣OOOOO",
            "␣OOO␣O␣␣␣O",
            "␣OOO␣O␣␣␣O",
            "␣OOO␣O␣␣␣O",
            "␣␣␣␣␣OOOOO"},
        {"O␣␣OOOO␣␣O",
            "OO␣␣OOOO␣␣",
            "OOO␣␣OOOO␣",
            "OOOO␣␣OOOO",
            "␣OOOO␣␣OOO",
            "␣␣OOOO␣␣OO",
            "O␣␣OOOO␣␣O",
            "OO␣␣OOOO␣␣",
            "OOO␣␣OOOO␣",
            "OOOO␣␣OOOO"},
        {"OOOOOOOOOO",
            "O␣␣␣␣␣␣␣␣O",
            "O␣␣␣␣␣␣␣␣O",
            "O␣␣␣␣␣␣␣␣O",
            "O␣␣␣OO␣␣␣O",
            "O␣␣␣OO␣␣␣O",
            "O␣␣␣␣␣␣␣␣O",
            "O␣␣␣␣␣␣␣␣O",
            "O␣␣␣␣␣␣␣␣O",
            "OOOOOOOOOO"},
        {"␣␣␣␣OO␣␣␣␣",
            "␣OOO␣␣OOO␣",
            "␣OO␣␣␣␣OO␣",
            "␣OO␣␣␣␣OO␣",
            "␣OO␣␣OOOO␣",
            "␣OOOOOOOO␣",
            "␣O␣␣O␣␣OO␣",
            "␣O␣O␣␣␣OO␣",
            "␣OOOOOOOO␣",
            "␣OOOOOOOO␣"},
           {"␣␣OOO␣␣␣␣␣",
            "␣OOOOOOOO␣",
            "OOO␣␣␣OOO␣",
            "␣OO␣␣␣␣OO␣",
            "␣OOOOOOOO␣",
            "␣OOO␣␣␣␣␣␣",
            "␣OO␣␣␣␣OO␣",
            "␣O␣␣␣␣O␣␣␣",
            "␣OO␣␣␣␣OO␣",
            "␣␣␣␣␣␣␣␣␣␣"}};

    public BiPolarNeuralData convertPattern(String[][] data, int index) {
        int resultIndex = 0;
        BiPolarNeuralData result = new BiPolarNeuralData(WIDTH * HEIGHT);
        for (int row = 0; row < HEIGHT; row++) {
            for (int col = 0; col < WIDTH; col++) {
                char ch = data[index][row].charAt(col);
                result.setData(resultIndex++, ch == 'O');
            }
        }
        return result;
    }

    public void display(BiPolarNeuralData pattern1, BiPolarNeuralData
        pattern2) {
        int index1 = 0;
        int index2 = 0;

        for (int row = 0; row < HEIGHT; row++) {
            StringBuilder line = new StringBuilder();

            for (int col = 0; col < WIDTH; col++) {
                if (pattern1.getBoolean(index1++)) {
                    line.append('O');
                } else {
                    line.append('␣');
                }
            }

            line.append("␣␣␣->␣␣␣");

            for (int col = 0; col < WIDTH; col++) {
                if (pattern2.getBoolean(index2++)) {
                    line.append('O');
                } else {
                    line.append('␣');
                }
            }



            System.out.println(line.toString());
        }
    }

    public void evaluate(HopfieldNetwork hopfieldLogic, String[][] pattern)
        {
        for (int i = 0; i < pattern.length; i++) {
            BiPolarNeuralData pattern1 = convertPattern(pattern, i);
            hopfieldLogic.setCurrentState(pattern1);
            int cycles = hopfieldLogic.runUntilStable(100);
            BiPolarNeuralData pattern2 = (BiPolarNeuralData) hopfieldLogic.
                getCurrentState();
            System.out.println("Cycles␣until␣stable(max␣100):␣" + cycles + "
                ,␣result=");
            display(pattern1, pattern2);
            System.out.println("----------------------");
        }
    }

    public void run() {
        /*HopfieldPattern pattern = new HopfieldPattern();
         pattern.setInputNeurons(WIDTH*HEIGHT);
         BasicNetwork hopfield = pattern.generate();
         HopfieldLogic hopfieldLogic = (HopfieldLogic)hopfield.getLogic();*/

        HopfieldNetwork hopfieldLogic = new HopfieldNetwork(WIDTH * HEIGHT);

        for (int i = 0; i < PATTERN.length; i++) {
            hopfieldLogic.addPattern(convertPattern(PATTERN, i));
        }

        evaluate(hopfieldLogic, PATTERN);
        evaluate(hopfieldLogic, PATTERN2);
    }
}
```

A.23 Package be.hogent.captchasolvingnetwork.network.en

Listing A.25: be.hogent.captchasolvingnetwork.network.NeuralNetwork

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchasolvingnetwork.network;

import java.io.Serializable;

/**
 * NeuralNetwork.java (UTF-8)
 *
 * Abstract class that all neural networks should extend, this is to
     streamline
 * the testing and building statics phase. The actions of the networks are
 * defined by NeuralNetworkActions interface implements serialisable for
     saving
 * the networks.
 *
 * 2013/05/19
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.1.2
 * @see NeuralNetworkActions
 */
public abstract class NeuralNetwork implements NeuralNetworkActions,
    Serializable {

    private int id, hSize, vSize;

    /**
     * Default constructor, sets the id to -1, hSize to 40 and vSize to 50.
     *
     */
    public NeuralNetwork() {
        this(-1, 40, 50);
    }

    /**
     * Constructor
     *
     * @param id the id of the network
     * @param hSize the horizontal size (width)
     * @param vSize the vertical size (height)
     */
    public NeuralNetwork(int id, int hSize, int vSize) {
        this.id = id;
        this.hSize = hSize;
        this.vSize = vSize;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getHsize() {
        return hSize;
    }

    public void setHsize(int hSize) {
        this.hSize = hSize;
    }

    public int getVsize() {
        return vSize;
    }

    public void setVsize(int vSize) {
        this.vSize = vSize;
    }

    public abstract String getLayerLayout();
}
```

Listing A.26: be.hogent.captchasolvingnetwork.network.NeuralNetworkActions

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchasolvingnetwork.network;

/**
 * NeuralNetworkActions.java (UTF-8)
 *
 * Interface that defines the actions all NeuralNetworks should implement
 *
 * 2013/05/20
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public interface NeuralNetworkActions {

    /**
     * Build/generates the network.
     */
    public void buildNetwork();

    /**
     * Trains the network
     */
    public void trainNetwork();

    /**
     * evaluates the input with the network.
     *
     * @param input the object to be evaluated
     * @param maxIterations the maximum iterations before giving up
     * @return the result
     */
    public double[] evaluate(double[] input, int maxIterations);
}
```

A.24 Package be.hogent.captchasolvingnetwork.network.neuropl

Listing A.27: be.hogent.captchasolvingnetwork.util.CharacterPatternUtils

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchasolvingnetwork.util;

import java.util.Arrays;

/**
 * CharacterPatternUtils.java (UTF-8)
 *
 * Utility class to for operations concerning network training and testing.
 *
 * 2013/05/20
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class CharacterPatternUtils {

    public static double[] characterToBitArray(char c) {
        String bitString = Integer.toBinaryString((int) c);
        System.err.println(c + " bitstring: " + bitString);

        // leftpad the string with 0 so it is atleast 8 bit long;
        while (bitString.length() < 8) {
            bitString = "0" + bitString;
        }

        double bit = 0;
        double[] result = new double[8];
        int resultIndex = 7;

        for (int i = result.length - 1; i > 0; i--) {
            if (bitString.charAt(i) == '1') {
                bit = 1;
            } else {
                bit = 0;
            }
            result[resultIndex--] = bit;
        }

        System.err.println(c + " bitArray: " + Arrays.toString(result));
        return result;
    }
}
```

Listing A.28: be.hogent.captchasolvingnetwork.util.EncogTrainingSet

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package be.hogent.bulksolvingstatistics.domain.neuralnetwork.encogutils;

import be.hogent.captchabuilder.elementcreator.renderer.text.
    AbstractWordRenderer;
import be.hogent.captchabuilder.elementcreator.renderer.text.
    DefaultWordRenderer;
import be.hogent.captchabuilder.elementcreator.renderer.text.WordRenderer;
import be.hogent.captchabuilder.util.ColorRangeRGBA;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import be.hogent.captchasolvingnetwork.util.CharacterPatternUtils;
import be.hogent.captchasolvingnetwork.util.ImageToInputPattern;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.geom.AffineTransform;
import java.awt.image.AffineTransformOp;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;

/**
 * EncogTrainingSet.java (UTF-8)
 *
 * Utility class to help generate the input and output trainingsets for an
 *   encog
 * Neural Network.
 *
 * 2013/05/20
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class EncogTrainingSet {

    public static double[][] buildTrainingInputSet(char[] chars, int hSize,
        int vSize) {
        double[][] inputTrainingsSet = new double[chars.length][];
        System.out.println("building␣Trainingsets");
        BufferedImage img;
        WordRenderer renderer = new DefaultWordRenderer(new ColorRangeRGBA
            (0, 0, 0, 255), AbstractWordRenderer.DEFAULT_FONTS, 0, 0.25,
            CaptchaConstants.DEFAULT_STROKE_WIDTH);
        int index = 0;

        for (char c : chars) {
            img = new BufferedImage(40, 50, BufferedImage.TYPE_INT_ARGB);
            renderer.render(String.valueOf(c), img);

            // check if size == the default size (40*50) if not scale
            if (hSize != 40 || vSize != 50) {
                BufferedImage resized = new BufferedImage(hSize, vSize, img.
                    getType());
                Graphics2D g = resized.createGraphics();
                g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
                    RenderingHints.VALUE_INTERPOLATION_BILINEAR);
                g.drawImage(img, 0, 0, hSize, vSize, 0, 0, img.getWidth(),
                    img.getHeight(), null);
                g.dispose();

                //replace the origal with the resized
                img = resized;
            }

            try {
                String path = "TrainingsetImages/";
                // if the directory does not exist, create it and it's
                    parents
                File theDir = new File(path);
                if (!theDir.exists()) {
                    System.out.println("creating␣directory:␣" + path);
                    boolean result = theDir.mkdirs();
                    if (result) {
                        System.out.println("Directory␣created");
                    }
                }

                ImageIO.write(img, "png", new File(path + Character.getName(
                    c) + "-" + hSize + "X" + vSize + ".png"));
            } catch (IOException ex) {
                System.err.println(ex.getMessage());
            }

            inputTrainingsSet[index++] = ImageToInputPattern.
                colorRangeToDoubleInputPattern(img, 0, 0);
        }

        return inputTrainingsSet;
    }

    public static double[][] buildTrainingIdealSet(char[] chars) {
        double[][] outputTrainingsSet = new double[chars.length][];
        System.out.println("building␣TrainingIdealSet");
        int index = 0;

        for (char c : chars) {
            outputTrainingsSet[index++] = CharacterPatternUtils.
                characterToBitArray(c);
        }

        return outputTrainingsSet;
    }
}
```

Listing A.29: be.hogent.captchasolvingnetwork.util.ImageToInputPattern

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchasolvingnetwork.util;

import be.hogent.captchacleanup.utils.ImageToArray;
import java.awt.image.BufferedImage;
import org.encog.ml.data.specific.BiPolarNeuralData;

/**
 * ImageToInputPattern.java (UTF-8)
 *
 * Utility class to convert an Image to a usable pattern for input to a
    network
 * This will reduce a 2-dimensional image to 1-dimensional array of doubles
 *
 * 2013/05/19
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class ImageToInputPattern {

    /**
     * reduce a 2-dimensional image to 1-dimensional array of doubles based
        on the colour range supplied.
     *
     * @param img the image to be transformed
     * @param startRange the numerical (!NOT HEX) value of the range start (
        inclusive)
     * @param endRange the numerical (!NOT HEX) value of the range end (
        inclusive)
     * @return the neural network input pattern based on the image.
     */
    public static double[] colorRangeToDoubleInputPattern(BufferedImage img,
        int startRange, int endRange) {
```

```
        return reduceDimension(ImageToArray.colorRangeToDoubleArray(img,
            startRange, endRange));
    }

    private static double[] reduceDimension(double[][] data) {
        int resultIndex = 0;
        double[] result = new double[data.length * data[0].length];
        for (int y = 0; y < data[0].length; y++) {
            for (int x = 0; x < data.length; x++) {
                result[resultIndex++] = data[x][y];
            }
        }
        return result;
    }

    public static BiPolarNeuralData colorRangeToBiPolarNeuralData(
        BufferedImage img, int startRange, int endRange) {
        return booleanArrayToBiPolarNeuralData(ImageToArray.
            colorRangeToBooleanArray(img, startRange, endRange));
    }

    private static BiPolarNeuralData booleanArrayToBiPolarNeuralData(boolean
        [][] data){
        int resultIndex = 0;
        int width = data.length;
        int height = data[0].length;
        BiPolarNeuralData result = new BiPolarNeuralData(width* height);
        for (int y = 0; y < height; y++) {
            for (int x = 0; x < width; x++) {
                result.setData(resultIndex++, data[x][y]);
            }
        }
        return result;
    }
}
```

A.25 Package be.hogent.bulksolvingstatistics.domain.neura

Listing A.30: be.hogent.bulksolvingstatistics.domain.neuralnetwork.DefaultNeuralNetworkControll

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.bulksolvingstatistics.domain.neuralnetwork;

import be.hogent.bulksolvingstatistics.domain.neuralnetwork.dataobjects.
    TestResultDataObjectBuilder;
import be.hogent.bulksolvingstatistics.persistance.PersistanceController;
import be.hogent.captchabuilder.builder.Captcha;
import be.hogent.captchabuilder.builder.CaptchaBuilder;
import be.hogent.captchasolvingnetwork.network.NeuralNetwork;
import be.hogent.captchasolvingnetwork.util.CharacterPatternUtils;
import be.hogent.captchasolvingnetwork.util.ImageToInputPattern;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.image.BufferedImage;
import java.sql.SQLException;
import java.util.Arrays;
import org.apache.commons.cli.ParseException;

/**
 * DefaultNeuralNetworkController.java (UTF-8)
 *
 * Default controller implementation.
 *
 * 2013/05/19
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class DefaultNeuralNetworkController implements
    NeuralNetworkController {

    private NeuralNetwork network;
    private TestResultDataObjectBuilder resultBuilder;

    public DefaultNeuralNetworkController() {
    }

    public DefaultNeuralNetworkController(NeuralNetwork network) {
        this();
        this.network = network;
    }

    @Override
    public NeuralNetwork getNetwork() {
        return network;
    }
    @Override
    public void setNetwork(NeuralNetwork network) {
        this.network = network;
    }

    @Override
    public void buildNetwork() {
        network.buildNetwork();
    }

    @Override
    public void trainNetwork() {
        network.trainNetwork();
    }

    @Override
    public double[] evaluate(double[] input, int maxIterations) {
        return network.evaluate(input, maxIterations);
    }

    @Override
    public void evaluate(String captchaBuildString, int amount, int
        maxIterations) {
        double[] input, result, expectedResult;
        CaptchaBuilder captchaBuilder;
        Captcha captcha;
        char c;
        BufferedImage img;

        for (int i = 0; i < amount; i++) {
            try {
                boolean correct;

                captchaBuilder = new CaptchaBuilder(40, 50,
                    captchaBuildString);
                captcha = captchaBuilder.buildCaptcha();
                c = captcha.getAnswer().charAt(0);
                img = captcha.getImage();

                // check if size == the default size (40*50) if not scale
                if (network.getHsize() != 40 || network.getVsize() != 50) {
                    BufferedImage resized = new BufferedImage(network.
                        getHsize(), network.getVsize(), img.getType());
                    Graphics2D g = resized.createGraphics();
                    g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
                        RenderingHints.VALUE_INTERPOLATION_BILINEAR);
                    g.drawImage(img, 0, 0, network.getHsize(), network.
                        getVsize(), 0, 0, img.getWidth(), img.getHeight(),
                        null);
                    g.dispose();

                    //replace the origal with the resized
                    img = resized;
                }

                input = ImageToInputPattern.colorRangeToDoubleInputPattern(
                    img, 0, 0);
                expectedResult = CharacterPatternUtils.characterToBitArray(c
                    );

                long startTimeLong = System.nanoTime();
                result = evaluate(input, maxIterations);
                long endTimeLong = System.nanoTime();
                double durationInSec = (double) ((endTimeLong -
                    startTimeLong) / Math.pow(10, 9));


                System.out.println("Processing output");
                for (int j = 0; j < result.length; j++) {
                    result[j] = (result[j] >= 0.5) ? 1 : 0;
                }

                if (Arrays.equals(result, expectedResult)) {
                    System.out.println(c + " recognized correctly");
                    correct = true;
                } else {
                    System.out.println(c + " recognized Incorrectly");
                    System.err.println("result: " + Arrays.toString(result)
                        + " != " + Arrays.toString(expectedResult));
                    correct = false;
                }

                //create the builder
                resultBuilder = new TestResultDataObjectBuilder();

                //set the network id (should be != -1 if set correctly by
                    saving
                resultBuilder.setNetworkID(network.getId())
                        .setCharacter(c + "")
                        .setTestType(captchaBuildString)
                        .setDuration(durationInSec)
                        .setCorrect(correct);

                PersistanceController.getInstance().addTestResult(
                    resultBuilder.createTestResultDataObject());

            } catch (ParseException | SQLException | ClassNotFoundException
                ex) {
                System.err.println(ex.getMessage());
            }
        }
    }
}
```

Listing A.31: be.hogent.bulksolvingstatistics.domain.neuralnetwork.DefaultNeuralNetworkRepository

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.bulksolvingstatistics.domain.neuralnetwork;

import be.hogent.bulksolvingstatistics.persistance.PersistanceController;
import be.hogent.captchasolvingnetwork.network.NeuralNetwork;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

/**
 * DefaultNeuralNetworkRepository.java (UTF-8)
 *
 * Default repository implementation.
 *
 * 2013/05/19
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class DefaultNeuralNetworkRepository implements
    NeuralNetworkRepository {

    private List<NeuralNetwork> repository;

    public DefaultNeuralNetworkRepository() {
        repository = new ArrayList<>();
    }

    public DefaultNeuralNetworkRepository(List<NeuralNetwork> repository) {
        this.repository = repository;
    }

    @Override
    public NeuralNetwork get(int id) {
        for (NeuralNetwork neuralNetwork : repository) {
            if (neuralNetwork.getId() == id) {
                return neuralNetwork;
            }
        }

        throw new IllegalArgumentException("Network with id: " + id + "not
            found.");
    }

    @Override
    public void add(NeuralNetwork network) {
        try {
        } catch (Exception e) {
            throw new IllegalArgumentException("The␣network␣could␣not␣be␣
                saved␣to␣the␣database.", e);
        }
        repository.add(network);
    }

    @Override
    public void remove(NeuralNetwork network) {
        try {

        } catch (Exception e) {
            throw new IllegalArgumentException("The␣network␣could␣not␣be␣
                removed␣from␣the␣database.", e);
        }
        repository.remove(network);
    }

    @Override
    public Collection<NeuralNetwork> all() {
        return repository;
    }
}
```

A.26 Package be.hogent.bulksolvingstatistics.domain.neuralnetw

Listing A.32: be.hogent.bulksolvingstatistics.domain.neuralnetwork.NeuralNetworkController

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.bulksolvingstatistics.domain.neuralnetwork;

import be.hogent.captchasolvingnetwork.network.NeuralNetwork;
import be.hogent.captchasolvingnetwork.network.NeuralNetworkActions;

/**
 * NeuralNetworkController.java (UTF-8)
 *
 * Interface defining the mandatory implemented functions.
 *
 * 2013/05/19
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public interface NeuralNetworkController extends NeuralNetworkActions {

    public NeuralNetwork getNetwork();

    public void setNetwork(NeuralNetwork network);

    public void evaluate(String captchaBuildString, int amount, int
        maxIterations);
}
```

Listing A.33: be.hogent.bulksolvingstatistics.domain.neuralnetwork.NeuralNetworkRepository

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.bulksolvingstatistics.domain.neuralnetwork;

import be.hogent.captchasolvingnetwork.network.NeuralNetwork;
import java.util.Collection;


/**
 * NeuralNetworkRepository.java (UTF-8)
 *
 * Interface defining the mandatory implemented functions.
 *
 * 2013/05/19
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public interface NeuralNetworkRepository {

    public Collection<NeuralNetwork> all();

    public NeuralNetwork get(int id);

    public void add(NeuralNetwork network);

    public void remove(NeuralNetwork network);
}
```

Listing A.34: be.hogent.bulksolvingstatistics.persistance.mappers.Mapper

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.bulksolvingstatistics.persistance.mappers;

import java.sql.SQLException;
import java.util.Collection;

/**
 * NeuralNetworkMapper.java (UTF-8)
 *
 * This interface defines the CRUD operations for mapping an object to
     database
 * records.
 *
 * 2013/05/20
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public interface Mapper<T> {

    public T add(T object) throws ClassNotFoundException, SQLException;

    public Collection<T> findAll() throws ClassNotFoundException,
        SQLException;

    public T find(int id) throws ClassNotFoundException, SQLException;

    public T upate(T object) throws ClassNotFoundException, SQLException;

    public void delete(T object) throws ClassNotFoundException, SQLException
        ;

}
```

Listing A.35: be.hogent.bulksolvingstatistics.persistance.mappers.NeuralNetworkMapper

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.bulksolvingstatistics.persistance.mappers;

import be.hogent.bulksolvingstatistics.domain.neuralnetwork.dataobjects.
    NeuralNetworkDataObject;
import be.hogent.bulksolvingstatistics.domain.neuralnetwork.dataobjects.
    NeuralNetworkDataObjectBuilder;
import be.hogent.bulksolvingstatistics.persistance.PersistanceController;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * NeuralNetworkMapper.java (UTF-8)
 *
 * This class maps NeuralNetworkDataObject to database records.
 *
 * 2013/05/20
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class NeuralNetworkMapper implements Mapper<NeuralNetworkDataObject>
    {

    private final String addStatement = "INSERT␣INTO␣networks␣(type,␣layout,
        ␣accuracy,␣trainingduration,␣iterations,␣networksavedlocation)␣
        VALUES␣(?,␣?,␣?,␣?,␣?,␣?)";
    private final String getStatement = "Select␣*␣FROM␣networks␣WHERE␣id=?";
    private final String getAllStatement = "Select␣*␣FROM␣networks";
    private final String updateStatement = "UPDATE␣networks␣SET␣type=?,␣
        layout=?,␣accuracy=?,␣trainingduration=?,␣iterations=?,␣
        networksavedlocation=?␣WHERE␣id=?";
    private final String deleteStatement = "DELETE␣FROM␣networks␣WHERE␣id=?"
        ;
    private PreparedStatement statement;
    private ResultSet resultSet;
    private Connection connection;

    @Override
    public NeuralNetworkDataObject add(NeuralNetworkDataObject object)
        throws SQLException, ClassNotFoundException {
        try {
            connection = PersistanceController.getInstance().getConnection()
                ;
            statement = connection.prepareStatement(addStatement, Statement.
                RETURN_GENERATED_KEYS);

            statement.setString(1, object.getNetworkType());
            statement.setString(2, object.getLayerLayout());
            statement.setDouble(3, object.getAccuracy());
            statement.setDouble(4, object.getTrainingDuration());
            statement.setInt(5, object.getIterations());
            statement.setString(6, object.getSavedLocation());

            int affectedRows = statement.executeUpdate();
            if (affectedRows == 0) {
                throw new SQLException("Creating␣network␣failed,␣no␣rows␣
                    affected.");
            }

            resultSet = statement.getGeneratedKeys();
            if (resultSet.next()) {
                object.setId(resultSet.getInt(1));
            } else {
                throw new SQLException("Creating␣network␣failed,␣no␣
                    generated␣key␣obtained.");
            }

            return object;
        } catch (SQLException | ClassNotFoundException ex) {
            Logger.getLogger(TestResultMapper.class.getName()).log(Level.
                SEVERE, null, ex);
            throw ex;
        } finally {
            if (resultSet != null) {
                try {
                    resultSet.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (statement != null) {
                try {
                    statement.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (connection != null) {
                try {
                    connection.close();
                } catch (SQLException logOrIgnore) {
                }
            }
        }
    }

    @Override
    public Collection<NeuralNetworkDataObject> findAll() throws
        ClassNotFoundException, SQLException {
        NeuralNetworkDataObjectBuilder builder;
        List<NeuralNetworkDataObject> coll = new ArrayList<>();
        try {
            statement = PersistanceController.getInstance().getConnection().
                prepareStatement(getAllStatement);
            resultSet = statement.executeQuery(getAllStatement);
            while (resultSet.next()) {
                builder = new NeuralNetworkDataObjectBuilder();
                builder.setId(resultSet.getInt("id"));
                builder.setNetworkType(resultSet.getString("type"));
                builder.setLayerLayout(resultSet.getString("layout"));
                builder.setAccuracy(resultSet.getDouble("accuracy"));
                builder.setTrainingDuration(resultSet.getDouble("
                    trainingduration"));
                builder.setIterations(resultSet.getInt("iterations"));
                builder.setSavedLocation(resultSet.getString("
                    networksavedlocation"));
            }
        } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(TestResultMapper.class.getName()).log(Level.
                SEVERE, null, ex);
            throw ex;
        } finally {
            if (resultSet != null) {
                try {
                    resultSet.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (statement != null) {
                try {
                    statement.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (connection != null) {
                try {
                    connection.close();
                } catch (SQLException logOrIgnore) {
                }
            }
        }

        return coll;
    }

    @Override
    public NeuralNetworkDataObject find(int id) throws
        ClassNotFoundException, SQLException {
        NeuralNetworkDataObjectBuilder builder;
        try {
            statement = PersistanceController.getInstance().getConnection().
                prepareStatement(getStatement);

            statement.setInt(1, id);

            resultSet = statement.executeQuery(getStatement);

            while (resultSet.next()) {
                builder = new NeuralNetworkDataObjectBuilder();
                builder.setId(resultSet.getInt("id"));
                builder.setNetworkType(resultSet.getString("type"));
                builder.setLayerLayout(resultSet.getString("layout"));
                builder.setAccuracy(resultSet.getDouble("accuracy"));
                builder.setTrainingDuration(resultSet.getDouble("
                    trainingduration"));
                builder.setIterations(resultSet.getInt("iterations"));
                builder.setSavedLocation(resultSet.getString("
                    networksavedlocation"));
                return builder.createNeuralNetworkDataObject();
            }

            throw new IllegalArgumentException("Network␣with␣ID:␣" + id + "␣
                not␣found");
        } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(TestResultMapper.class.getName()).log(Level.
                SEVERE, null, ex);
            throw ex;
        } finally {
            if (resultSet != null) {
                try {
                    resultSet.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (statement != null) {
                try {
                    statement.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (connection != null) {
                try {
                    connection.close();
                } catch (SQLException logOrIgnore) {
                }
            }
        }
    }

    @Override
    public NeuralNetworkDataObject upate(NeuralNetworkDataObject object)
        throws SQLException, ClassNotFoundException {
        try {
            connection = PersistanceController.getInstance().getConnection()
                ;
            statement = connection.prepareStatement(updateStatement);

            statement.setString(1, object.getNetworkType());
            statement.setString(2, object.getLayerLayout());
            statement.setDouble(3, object.getAccuracy());
            statement.setDouble(4, object.getTrainingDuration());
            statement.setInt(5, object.getIterations());
            statement.setString(6, object.getSavedLocation());
            statement.setInt(7, object.getId());

            int affectedRows = statement.executeUpdate();
            if (affectedRows == 0) {
                throw new SQLException("updating␣network␣failed,␣no␣rows␣
                    affected.");
            }

            return object;
        } catch (SQLException | ClassNotFoundException ex) {
            Logger.getLogger(TestResultMapper.class.getName()).log(Level.
                SEVERE, null, ex);
            throw ex;
        } finally {
            if (resultSet != null) {
                try {
                    resultSet.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (statement != null) {
                try {
                    statement.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (connection != null) {
                try {
                    connection.close();
                } catch (SQLException logOrIgnore) {
                }
            }
        }
    }

    @Override
    public void delete(NeuralNetworkDataObject object) throws SQLException,
        ClassNotFoundException {
        try {
            connection = PersistanceController.getInstance().getConnection()
                ;
            statement = connection.prepareStatement(deleteStatement);

            statement.setInt(1, object.getId());

            int affectedRows = statement.executeUpdate();
            if (affectedRows == 0) {
                throw new SQLException("deleting␣network␣failed,␣no␣rows␣
                    affected.");
            }

        } catch (SQLException | ClassNotFoundException ex) {
            Logger.getLogger(TestResultMapper.class.getName()).log(Level.
                SEVERE, null, ex);
            throw ex;
        } finally {
            if (resultSet != null) {
                try {
                    resultSet.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (statement != null) {
                try {
                    statement.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (connection != null) {
                try {
                    connection.close();
                } catch (SQLException logOrIgnore) {
                }
            }
        }
    }
}
```

Listing A.36: be.hogent.bulksolvingstatistics.persistance.mappers.TestResultMapper

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.bulksolvingstatistics.persistance.mappers;

import be.hogent.bulksolvingstatistics.domain.neuralnetwork.dataobjects.
    TestResultDataObject;
import be.hogent.bulksolvingstatistics.domain.neuralnetwork.dataobjects.
    TestResultDataObjectBuilder;
import be.hogent.bulksolvingstatistics.persistance.PersistanceController;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * DatabaseConnection.java (UTF-8)
 *
 * This class maintains the connection between the application and the
     SQLite
 * database.
 *
 * 2013/05/20
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class TestResultMapper implements Mapper<TestResultDataObject> {

    private final String addStatement = "INSERT␣INTO␣tests␣(network_id,␣
        testtype,␣character,␣duration,␣correct)␣VALUES␣(?,␣?,␣?,␣?,␣?)";
    private final String getStatement = "Select␣*␣FROM␣tests␣WHERE␣id=?";
    private final String getAllStatement = "Select␣*␣FROM␣tests";
    private final String updateStatement = "UPDATE␣tests␣SET␣network_id=?,␣
        testtype=?,␣character=?,␣duration=?,␣correct=?␣WHERE␣id=?";
    private final String deleteStatement = "DELETE␣FROM␣tests␣WHERE␣id=?";
    private PreparedStatement statement;
    private ResultSet resultSet;
    private Connection connection;

    @Override
    public TestResultDataObject add(TestResultDataObject object) {
        try {
            connection = PersistanceController.getInstance().getConnection()
                ;
            statement = connection.prepareStatement(addStatement, Statement.
                RETURN_GENERATED_KEYS);

            statement.setInt(1, object.getNetworkID());
            statement.setString(2, object.getTestType());
            statement.setString(3, object.getCharacter());
            statement.setDouble(4, object.getDuration());
            statement.setBoolean(5, object.isCorrect());

            int affectedRows = statement.executeUpdate();
            if (affectedRows == 0) {
                throw new SQLException("Creating␣testrecord␣failed,␣no␣rows␣
                    affected.");
            }

            resultSet = statement.getGeneratedKeys();
            if (resultSet.next()) {
                object.setId(resultSet.getInt(1));
            } else {
                throw new SQLException("Creating␣testrecord␣failed,␣no␣
                    generated␣key␣obtained.");
            }

            return object;
        } catch (SQLException | ClassNotFoundException ex) {
            Logger.getLogger(TestResultMapper.class.getName()).log(Level.
                SEVERE, null, ex);
        } finally {
            if (resultSet != null) {
                try {
                    resultSet.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (statement != null) {
                try {
                    statement.close();
                } catch (SQLException logOrIgnore) {
            }
        }
        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException logOrIgnore) {
            }
        }
    }

    return null;
}

@Override
public Collection<TestResultDataObject> findAll() throws
    ClassNotFoundException, SQLException {
    TestResultDataObjectBuilder builder;
    List<TestResultDataObject> coll = new ArrayList<>();
    try {
        statement = PersistanceController.getInstance().getConnection().
            prepareStatement(getAllStatement);
        resultSet = statement.executeQuery(getAllStatement);

        while (resultSet.next()) {
            builder = new TestResultDataObjectBuilder();
            builder.setID(resultSet.getInt("id"));
            builder.setCharacter(resultSet.getString("character"));
            builder.setCorrect(resultSet.getBoolean("correct"));
            builder.setDuration(resultSet.getDouble("duration"));
            builder.setNetworkID(resultSet.getInt("network_id"));
            builder.setTestType(resultSet.getString("type"));
            coll.add(builder.createTestResultDataObject());
        }
    } catch (ClassNotFoundException | SQLException ex) {
        Logger.getLogger(TestResultMapper.class.getName()).log(Level.
            SEVERE, null, ex);
        throw ex;
    } finally {
        if (resultSet != null) {
            try {
                resultSet.close();
            } catch (SQLException logOrIgnore) {
            }
        }
        if (statement != null) {
            try {
                statement.close();
            } catch (SQLException logOrIgnore) {
            }
        }
        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException logOrIgnore) {
            }
        }
    }

    return coll;
}

@Override
```

```
    public TestResultDataObject find(int id) throws ClassNotFoundException ,
        SQLException {
        TestResultDataObjectBuilder builder;
        try {
            statement = PersistanceController.getInstance().getConnection().
                prepareStatement(getStatement);

            statement.setInt(1, id);

            resultSet = statement.executeQuery(getStatement);

            while (resultSet.next()) {
                builder = new TestResultDataObjectBuilder();
                builder.setID(resultSet.getInt("id"));
                builder.setCharacter(resultSet.getString("character"));
                builder.setCorrect(resultSet.getBoolean("correct"));
                builder.setDuration(resultSet.getDouble("duration"));
                builder.setNetworkID(resultSet.getInt("network_id"));
                builder.setTestType(resultSet.getString("type"));
                return builder.createTestResultDataObject();
            }

            throw new IllegalArgumentException("Test␣result␣with␣ID:␣" + id
                + "␣not␣found");
        } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(TestResultMapper.class.getName()).log(Level.
                SEVERE, null , ex);
            throw ex;
        } finally {
            if (resultSet != null) {
                try {
                    resultSet.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (statement != null) {
                try {
                    statement.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (connection != null) {
                try {
                    connection.close();
                } catch (SQLException logOrIgnore) {
                }
            }
        }
    }

    @Override
    public TestResultDataObject upate(TestResultDataObject object) throws
        SQLException , ClassNotFoundException {
        try {
            connection = PersistanceController.getInstance().getConnection()
                ;
            statement = connection.prepareStatement(updateStatement);

            statement.setInt(1, object.getNetworkID());
            statement.setString(2, object.getTestType());
            statement.setString(3, object.getCharacter());
            statement.setDouble(4, object.getDuration());
            statement.setBoolean(5, object.isCorrect());
            statement.setInt(6, object.getId());

            int affectedRows = statement.executeUpdate();
            if (affectedRows == 0) {
                throw new SQLException("updating␣testrecord␣failed,␣no␣rows␣
                    affected.");
            }

            return object;
        } catch (SQLException | ClassNotFoundException ex) {
            Logger.getLogger(TestResultMapper.class.getName()).log(Level.
                SEVERE, null, ex);
            throw ex;
        } finally {
            if (resultSet != null) {
                try {
                    resultSet.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (statement != null) {
                try {
                    statement.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (connection != null) {
                try {
                    connection.close();
                } catch (SQLException logOrIgnore) {
                }
            }
        }
    }

    @Override
    public void delete(TestResultDataObject object) throws
        ClassNotFoundException, SQLException {
        try {
            connection = PersistanceController.getInstance().getConnection()
                ;
            statement = connection.prepareStatement(deleteStatement);

            statement.setInt(1, object.getId());

            int affectedRows = statement.executeUpdate();
            if (affectedRows == 0) {
                throw new SQLException("deleting␣testrecord␣failed,␣no␣rows␣
                    affected.");
            }

        } catch (SQLException | ClassNotFoundException ex) {
            Logger.getLogger(TestResultMapper.class.getName()).log(Level.
                SEVERE, null, ex);
            throw ex;
        } finally {
            if (resultSet != null) {
                try {
                    resultSet.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (statement != null) {
                try {
                    statement.close();
                } catch (SQLException logOrIgnore) {
                }
            }
            if (connection != null) {
                try {
                    connection.close();
                } catch (SQLException logOrIgnore) {
                }
            }
        }
    }
}
```

A.27 Package be.hogent.captchabuilder.elementcreator.produce

A.28 Package be.hogent.captchabuilder.elementcreator.produce

A.29 Package be.hogent.captchabuilder.elementcreator.produce

A.30 Package be.hogent.captchabuilder.elementcreator.produce

A.31 Package be.hogent.captchabuilder.elementcreator.renderer

A.32 Package be.hogent.captchabuilder.elementcreator.renderer

Listing A.37: be.hogent.captchabuilder.util.enums.CaptchaConstants

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.util.enums;

import java.security.SecureRandom;
import java.util.Random;

/**
 * CaptchaConstants.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.7
 * @version 1.0.7
 */
public class CaptchaConstants {

    public static final Random RANDOM = new SecureRandom();
    public static final char[] LETTERS = new char[]{'a', 'b', 'c', 'd', 'e',
         'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's
         ', 't', 'u', 'v', 'w', 'x', 'y', 'z'};
    public static final char[] NUMBERS = new char[]{'0', '1', '2', '3', '4',
         '5', '6', '7', '8', '9'};
    public static final char[] SPECIAL = new char[]{'&', '!', '@', '?', '#',
         '$', '%', '+', '='};
    public static final char[] REDUCEDALPHANUMERIC = new char[]{'a', 'b', 'c
         ', 'd', 'e', 'f', 'g', 'h', 'k', 'm', 'n', 'p', 'r', 'w', 'x', 'y',
         '2', '3', '4', '5', '6', '7', '8',};
    public static final char[] ARABIC_CHARS = {'\u0627', '\u0628', '\u062a',
         '\u062b', '\u062c', '\u062d', '\u062e', '\u062f', '\u0630', '\u0631
         ', '\u0632', '\u0633', '\u0634', '\u0635', '\u0636', '\u0637', '\
        u0638', '\u0639', '\u063a', '\u0641', '\u0642', '\u0643', '\u0644',
        '\u0645', '\u0646', '\u0647', '\u0648', '\u064a'};
    public static final int DEFAULT_LENGTH = 5;
    public static final double DEFAULT_YOFFSET = 0.25;
    public static final double DEFAULT_XOFFSET = 0.05;
    public static final float DEFAULT_STROKE_WIDTH = 0f;
    public static final String buildSequencelvl1Delim = "[:]+";
    public static final String buildSequencelvl2Delim = "[!]+";
    public static final String buildSequencelvl3Delim = "[#]+";
    public static final String buildSequencelvl4Delim = "[@]+";
    public static final String buildSequencelvl5Delim = "[*]+";
    public static final String buildSequencelvl6Delim = "[.]+";
    public static final String buildSequencelvl7Delim = "[?]+";
}
```

A.33 Package be.hogent.captchabuilder.util.enums.producer

A.34 Package be.hogent.captchabuilder.util.enums.renderer

Listing A.38: be.hogent.captchacleanup.utils.textfromimage.GetImageText

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package be.hogent.captchacleanup.utils.textfromimage;

//import com.sun.image.codec.jpeg.JPEGCodec;
//import com.sun.image.codec.jpeg.JPEGImageEncoder;
import java.awt.image.BufferedImage;
import java.io.File;
import java.util.LinkedList;
import javax.imageio.ImageIO;

/**
 *
 * @author Pieter
 */
public class GetImageText {

    private BufferedImage image;

    /**
     * Default constructor
     *
     * @param img The image containing text
     */
    public GetImageText(BufferedImage img) {
        image = img;
        merge_densityFactor = 0.5;
        merge_mass = 15;
        merge_dist1 = 4;
        merge_distfac = 1;
        merge_dist2 = 20;
    }

    /**
     * Constructor for testing purposes
     */
    public GetImageText(BufferedImage img, double m_densityFactor,
            int m_mass, int m_dist1, double m_distfac,
            int m_dist2) {
        image = img;
        merge_densityFactor = m_densityFactor;
        merge_mass = m_mass;
        merge_dist1 = m_dist1;
        merge_distfac = m_distfac;
        merge_dist2 = m_dist2;
    }

    /**
     * Only for debugging - prints out the current parameters
     */
    public void print() {
        System.out.println("m_densityFactor = " + merge_densityFactor);
        System.out.println("m_mass = " + merge_mass);
        System.out.println("m_dist1 = " + merge_dist1);
        System.out.println("m_distfac = " + merge_distfac);
        System.out.println("m_dist2 = " + merge_dist2);
    }

    int red(int rgb) {
        return (rgb & 0xff0000) >> 16;
    }

    int green(int rgb) {
        return (rgb & 0x00ff00) >> 8;
    }

    int blue(int rgb) {
        return rgb & 0xff;
    }

    int rgb(int red, int green, int blue) {
        return blue + (green << 8) + (red << 16);
    }

    /**
     * Discard boxes that do not appear to contain text
     */
    LinkedList discardNonText(LinkedList boxes, int[][] contrast) {
        int i = 0;
        while (i < boxes.size()) {
            int numberOfStems = 0;
            TextRegion thisBox = (TextRegion) boxes.get(i);
            // Count the stems in this box
            if (thisBox.y1 != thisBox.y2) {
                for (int a = thisBox.x1 + 1; a < thisBox.x2 - 1; a++) {
                    int thisStemHeight = 0;
                    for (int b = thisBox.y1 + 1; b < thisBox.y2 - 1; b++) {
                        if ((contrast[a][b] != 0
                                || contrast[a - 1][b] != 0
                                || contrast[a + 1][b] != 0)
                               && (contrast[a][b - 1] != 0
                                || contrast[a - 1][b - 1] != 0
                                || contrast[a + 1][b - 1] != 0)
                               && (contrast[a][b + 1] != 0
                                || contrast[a - 1][b + 1] != 0
                                || contrast[a + 1][b + 1] != 0)) {
                            thisStemHeight++;
                        }
                    }
                    //a stem must cover at least 70% of a vertical line
                    if ((100 * thisStemHeight) / thisBox.height() > 70) {
                        numberOfStems++;
                    }
                }
            }
            if (thisBox.area() < 50
                    || thisBox.aspect() > .2
                    || thisBox.height() < 5
                    || thisBox.width() < 20
                    // expect at least one stem for every <height> of <width
                        >
                    || numberOfStems < thisBox.width() / thisBox.height()) {
                boxes.remove(i--);
            }
            i++;
        }
        return (boxes);
    }

    /**
     * Shrink each box as much as possible
     */
    LinkedList shrink(LinkedList boxes, int[][] contrast) {
        int i = 0;
        while (i < boxes.size()) {
            TextRegion thisBox = (TextRegion) boxes.get(i);
            if (thisBox.x1 != thisBox.x2
                    && thisBox.y1 != thisBox.y2) {
                int total = 0;
                for (int a = thisBox.x1; a < thisBox.x2; a++) {
                    for (int b = thisBox.y1; b < thisBox.y2; b++) {
                        total += contrast[a][b];
                    }
                }
                double averagex = total / thisBox.height();
                double averagey = total / thisBox.width();
                int newx1 = thisBox.x1;
                int newx2 = thisBox.x2;
                int newy1 = thisBox.y1;
                int newy2 = thisBox.y2;
                boolean moved = true;
                while (newx1 < newx2 && moved) {
                    moved = false;
                    int t1 = 0, t2 = 0;
                    for (int b = thisBox.y1; b < thisBox.y2; b++) {
                        t1 += contrast[newx1][b];
                        t2 += contrast[newx2][b];
                    }
                    if (t1 < averagey) {
                        newx1++;
                        moved = true;
                    }
                    if (t2 < averagey) {
                        newx2--;
                        moved = true;
                    }
                }
                moved = true;
                while (newy1 < newy2 && moved) {
                    moved = false;
                    int t1 = 0, t2 = 0;
                    for (int a = thisBox.x1; a < thisBox.x2; a++) {
                        t1 += contrast[a][newy1];
                        t2 += contrast[a][newy2];
                    }
                    if (t1 < averagex) {
                        newy1++;
                        moved = true;
                    }
                    if (t2 < averagex) {
                        newy2--;
                        moved = true;
                    }
                }
                thisBox.x1 = newx1;
                thisBox.x2 = newx2;
                thisBox.y1 = newy1;
                thisBox.y2 = newy2;
            }
            i++;
        }
        return (boxes);
    }
    public double merge_densityFactor;
    public int merge_mass;
    public int merge_dist1;
    public double merge_distfac;
    public int merge_dist2;

    LinkedList merge(LinkedList boxes) {
        boolean change = true;
        while (change == true) {
            change = false;
            int i = 0;
            while (i < boxes.size()) {
                int j = 0;
                while (i < boxes.size() && j < boxes.size()) {
                    if (i != j) {
                        TextRegion thisBox = (TextRegion) boxes.get(i);
                        TextRegion thatBox = (TextRegion) boxes.get(j);
                        change = merge(thisBox, thatBox);
                        if (change) {
                            boxes.set(i, thisBox);
                            boxes.remove(j);
                            j--;
                        }
                    }
                    j++;
                }
                i++;
            }
        }
        return (boxes);
    }

    boolean merge(TextRegion thisBox, TextRegion thatBox) {
        int mergex1 = Math.min(thisBox.x1, thatBox.x1);
        int mergex2 = Math.max(thisBox.x2, thatBox.x2);
        int mergey1 = Math.min(thisBox.y1, thatBox.y1);
        int mergey2 = Math.max(thisBox.y2, thatBox.y2);
        double mergemass = thisBox.mass + thatBox.mass;
        double mergedensity = mergemass
                / ((mergex2 - mergex1) * (mergey2 - mergey1));
        double mergeaspect = ((double) mergey2 - mergey1) / ((double)
            mergex2 - mergex1);

        double reasonsToMerge = 0;
        if (mergedensity > merge_densityFactor * thisBox.density()) {
            reasonsToMerge++;
        }
        if (mergedensity > merge_densityFactor * thatBox.density()) {
            reasonsToMerge++;
        }
        if (mergeaspect < thisBox.aspect()) {
            reasonsToMerge++;
        }
        if (mergeaspect < thatBox.aspect()) {
            reasonsToMerge++;
        }
        if (thisBox.mass > merge_mass && thatBox.mass > merge_mass) {
            reasonsToMerge++;
        }
        int maxboxwidth = Math.max(thisBox.width(), thatBox.width());
        if (Math.abs(thisBox.y1 - thatBox.y1) < merge_dist1
                && Math.abs(thisBox.y2 - thatBox.y1) < merge_dist1
                && (Math.abs(thisBox.x1 - thatBox.x2) < merge_distfac *
                    maxboxwidth
                || Math.abs(thisBox.x2 - thatBox.x1)
                < merge_distfac * maxboxwidth)) {
            reasonsToMerge++;
        }
        if ((Math.abs(thisBox.y1 - thatBox.y1) < merge_dist2
                || Math.abs(thisBox.y2 - thatBox.y2) < merge_dist2)
                && (Math.abs(thisBox.x1 - thatBox.x2) < merge_distfac *
                    maxboxwidth
                || Math.abs(thisBox.x2 - thatBox.x1)
                < merge_distfac * maxboxwidth)) {
            reasonsToMerge++;
        }
        if (reasonsToMerge > 3) { // 7 reasons max
            thisBox.x1 = mergex1;
            thisBox.x2 = mergex2;
            thisBox.y1 = mergey1;
            thisBox.y2 = mergey2;
            thisBox.mass = mergemass;
            return true;
        }
        return false;
    }

    int[][] getContrast() {
        // Find pixels that stand out from the background
        int[][] contrast = new int[image.getWidth()][image.getHeight()];
        int[][] temp = new int[image.getWidth()][image.getHeight()];
        for (int i = 2; i < image.getWidth() - 2; i++) {
            for (int j = 2; j < image.getHeight() - 2; j++) {
                int thisPixel = image.getRGB(i, j);
                int left = image.getRGB(i - 1, j);
                int left2 = image.getRGB(i - 2, j);
                int right = image.getRGB(i + 1, j);
                int right2 = image.getRGB(i + 2, j);
                int up = image.getRGB(i, j - 1);
                int down = image.getRGB(i, j + 1);
                int t1 = 60; // thresholds
                int t2 = 80;
                if (Math.abs(blue(thisPixel) - blue(right)) > t1
                        || Math.abs(blue(thisPixel) - blue(left)) > t1
                        || Math.abs(blue(thisPixel) - blue(down)) > t1
                        || Math.abs(blue(thisPixel) - blue(up)) > t1
                        || Math.abs(blue(thisPixel) - blue(right2)) > t2
                        || Math.abs(blue(thisPixel) - blue(left2)) > t2
                        || Math.abs(green(thisPixel) - green(right)) > t1
                        || Math.abs(green(thisPixel) - green(left)) > t1
                        || Math.abs(green(thisPixel) - green(down)) > t1
                        || Math.abs(green(thisPixel) - green(up)) > t1
                        || Math.abs(green(thisPixel) - green(right2)) > t2
                        || Math.abs(green(thisPixel) - green(left2)) > t2
                        || Math.abs(red(thisPixel) - red(right)) > t1
                        || Math.abs(red(thisPixel) - red(left)) > t1
                        || Math.abs(red(thisPixel) - red(down)) > t1
                        || Math.abs(red(thisPixel) - red(up)) > t1
                        || Math.abs(red(thisPixel) - red(right2)) > t2
                        || Math.abs(red(thisPixel) - red(left2)) > t2) {
                    temp[i][j] = 1;
                }
            }
        }
        // Look for areas of contrast that extend vertically and
            horizontally
        // but not too far, to eliminate long straight lines (e.g. borders)
        for (int j = 2; j < image.getHeight() - 2; j++) {
            for (int i = 2; i < image.getWidth() - 2; i++) {
                if (temp[i][j] == 1) {
                    int width = 0;
                    int height = 0;
                    for (int k = 0;
                            i + k < image.getWidth() - 2
                            && i - k > 2
                            && (temp[i + k][j] == 1 || temp[i - k][j] == 1)
                            && width++ < 100;
                            k++)
                        ;
                    for (int k = 0;
                            j + k < image.getHeight() - 2
                            && j - k > 2
                            && (temp[i][j + k] == 1 || temp[i][j - k] == 1)
                            && height++ < 100;
                            k++)
                        ;
                    int totalOnLine = 0;
                    for (int k = Math.max(2, i - 40);
                            k < Math.min(image.getWidth() - 2, i + 40);
                            k++) {
                        totalOnLine += temp[k][j];
                    }
                    if (totalOnLine > 7 && width < 100 && height < 100) {
                        contrast[i][j] = 1;
                    }
                }
            }
        }
        return contrast;
    }

    /**
     * Looks for areas of text in an image.
     *
     * @return a LinkedList of boxes that are likely to contain text.
     */
    public LinkedList getTextBoxes() {
        LinkedList boxes = new LinkedList();

        int[][] contrast = getContrast();

        try {
            BufferedImage contrastpng = new BufferedImage(image.getWidth(),
                image.getHeight(), BufferedImage.TYPE_INT_RGB);
            for (int i = 0; i < image.getWidth(); i++) {
                for (int j = 0; j < image.getHeight(); j++) {
                    contrastpng.setRGB(i, j, 0xffffff * contrast[i][j]);
                }
            }
        } catch (Exception e) {
            System.out.println("Exception: " + e);
        }

        int contrastOnLine[] = new int[image.getHeight()];
        for (int j = 1; j < image.getHeight() - 1; j++) {
            int count = 0;
            contrastOnLine[j] = 0;
            for (int a = 0; a < image.getWidth(); a++) {
                count += contrast[a][j];
                contrastOnLine[j] += contrast[a][j];
            }
        }
        for (int j = 1; j < image.getHeight() - 1; j++) {
            contrastOnLine[j] = (contrastOnLine[j - 1]
                    + contrastOnLine[j]
                    + contrastOnLine[j + 1]) / 3;
        }
        for (int j = 1; j < image.getHeight() - 1; j++) {
            contrastOnLine[j] = (contrastOnLine[j - 1]
                    + contrastOnLine[j]
                    + contrastOnLine[j + 1]) / 3;
        }
        int averageOnLine = 0;
        for (int j = 1; j < image.getHeight() - 1; j++) {
            averageOnLine += contrastOnLine[j];
        }
        averageOnLine /= (image.getHeight() - 2);
        boolean intext = false;
        int boxstart = 0;
        int boxaverage = 0;
        int boxlines = 0;
        for (int j = 1; j < image.getHeight() - 1; j++) {
            if (contrastOnLine[j] > averageOnLine && !intext) {
                intext = true;
                boxstart = j;
                boxaverage = contrastOnLine[j];
                boxlines = 1;
            } else if (contrastOnLine[j] > averageOnLine) {
                boxaverage += contrastOnLine[j];
                boxlines++;
            } else if (contrastOnLine[j] <= averageOnLine && intext) {
                // found vertical limits, now find horizontal.
                intext = false;
                int boxend = j;
                if (boxend - boxstart > 10) {
                    // text must be higher than 10 pixels
                    boxaverage /= boxlines;
                    int contrastOnColumn[] = new int[image.getWidth()];
                    for (int i = 1; i < image.getWidth() - 1; i++) {
                        for (int b = boxstart; b < boxend; b++) {
                            contrastOnColumn[i] += contrast[i][b];
                        }
                    }
                    for (int i = 1; i < image.getWidth() - 1; i++) {
                        contrastOnColumn[i] = (contrastOnColumn[i - 1]
                                + contrastOnColumn[i]
                                + contrastOnColumn[i + 1]) / 3;
                    }
                    for (int i = 1; i < image.getWidth() - 1; i++) {
                        contrastOnColumn[i] = (contrastOnColumn[i - 1]
                                + contrastOnColumn[i]
                                + contrastOnColumn[i + 1]) / 3;
                    }
                    int averageOnColumn = 0;
                    for (int i = 1; i < image.getWidth() - 1; i++) {
                        averageOnColumn += contrastOnColumn[i];
                    }
                    averageOnColumn /= (image.getWidth() - 2);
                    boolean intextx = false;
                    int boxstartx = 0;
                    for (int i = 1; i < image.getWidth() - 1; i++) {
                        if (contrastOnColumn[i] > averageOnColumn / 2
                                && !intextx) {
                            intextx = true;
                            boxstartx = i;
                        } else if (contrastOnColumn[i] <= averageOnColumn /
                            2
                                && intextx) {
                            intextx = false;
                            int boxendx = i;
                            // found horizontal limits,
                            // now (if necessary) shrink
                            // vertical limits
                            int newcount = 0;
                            int tempboxstart = boxstart;
                            int tempboxend = boxend;
                            while (tempboxstart < boxend
                                    && newcount == 0) {
                                for (int a = boxstartx; a < boxendx; a++) {
                                    newcount += contrast[a][tempboxstart];
                                }
                                if (newcount < 2) {
                                    tempboxstart++;
                                }
                            }
                            newcount = 0;
                            while (tempboxstart < boxend && newcount == 0) {
                                for (int a = boxstartx; a < boxendx; a++) {
                                    newcount += contrast[a][tempboxend];
                                }
                                if (newcount < 2) {
                                    tempboxend--;
                                }
                            }
                            TextRegion thisBox = new TextRegion(boxstartx,
                                    tempboxstart,
                                    boxendx,
                                    tempboxend,
                                    image.getWidth(),
                                    image.getHeight(),
                                    boxaverage);
                            boxes.add(thisBox);
                        }
                    }
                }
            }
        }

        System.out.println(boxes.size() + " bounding boxes");
        shrink(boxes, contrast);
        boxes = merge(boxes);
        //shrink(boxes, contrast);
        System.out.println(boxes.size() + " bounding boxes after merge");
        boxes = discardNonText(boxes, contrast);
        System.out.println(boxes.size() + " bounding boxes after delete");
        return (shrink(boxes, contrast));
    }

    /**
     * Isolate text
     *
     * @return a <code>BufferedImage</code> value
     */
    public BufferedImage isolateText(LinkedList boxes) {
        BufferedImage outputimage = new BufferedImage(image.getWidth(),
                image.getHeight(),
                BufferedImage.TYPE_INT_RGB);
        // make everything monochrome
        for (int a = 0; a < image.getWidth(); a++) {
            for (int b = 0; b < image.getHeight(); b++) {
                int colour = image.getRGB(a, b);
                int average = (red(colour) + green(colour) + blue(colour)) /
                     3;
                outputimage.setRGB(a, b, rgb(average, average, average));
            }
        }
        // fill text boxes with colour
        for (int i = 0; i < boxes.size(); i++) {
            TextRegion thisBox = (TextRegion) boxes.get(i);
            int x1 = Math.max(1, thisBox.x1);
            int x2 = Math.min(image.getWidth() - 2, thisBox.x2);
            int y1 = Math.max(1, thisBox.y1);
            int y2 = Math.min(image.getHeight() - 2, thisBox.y2);
            for (int a = x1; a < x2; a++) {
                for (int b = y1; b < y2; b++) {
                    outputimage.setRGB(a, b, image.getRGB(a, b));
                }
            }
        }
        // draw red border around each text box
        int RED = 0xff0000;
        for (int i = 0; i < boxes.size(); i++) {
            TextRegion thisBox = (TextRegion) boxes.get(i);
            int x1 = Math.max(1, thisBox.x1);
            int x2 = Math.min(image.getWidth() - 2, thisBox.x2);
            int y1 = Math.max(1, thisBox.y1);
            int y2 = Math.min(image.getHeight() - 2, thisBox.y2);
            for (int a = x1; a < x2; a++) {
                outputimage.setRGB(a, thisBox.y1, RED);
                outputimage.setRGB(a, thisBox.y2, RED);
            }
            for (int a = y1; a < y2; a++) {
                outputimage.setRGB(thisBox.x1, a, RED);
                outputimage.setRGB(thisBox.x2, a, RED);
            }
        }
        return (outputimage);
    }
}
```

Listing A.39: be.hogent.captchacleanup.utils.textfromimage.TextRegion

```
/*
```

```
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package be.hogent.captchacleanup.utils.textfromimage;

/**
 *
 * @author Pieter
 */
public class TextRegion {
    int x1;
    int y1;
    int x2;
    int y2;
    double mass;

    /**
     * Creates a new <code>TextRegion</code> instance.
     *
     * @param xs an <code>int</code> value
     * @param ys an <code>int</code> value
     * @param xe an <code>int</code> value
     * @param ye an <code>int</code> value
     * @param maxx an <code>int</code> value
     * @param maxy an <code>int</code> value
     */
    TextRegion(int xs, int ys, int xe, int ye, int maxx, int maxy, double m)
         {
        if (xs < 0)
            x1 = 0;
        else if (xs > maxx)
            x1 = maxx;
        else x1 = xs;
        if (xe < 0)
            x2 = 0;
        else if (xe > maxx)
            x2 = maxx;
        else x2 = xe;
        if (ys < 0)
            y1 = 0;
        else if (ys > maxy)
            y1 = maxy;
        else y1 = ys;
        if (ye < 0)
            y2 = 0;
        else if (ye > maxy)
            y2 = maxy;
        else y2 = ye;
        mass = m;
    }

    int area() {
        return width() * height();
    }

    int height() {
        return y2 - y1;
    }

    int width() {
        return x2 - x1;
    }
    double density() {
        return mass / area();
    }

    double aspect() {
        return (double)height() / (double)width();
    }
}
```

Listing A.40: be.hogent.captchasolvingnetwork.network.encog.EncogBasicNetwork

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchasolvingnetwork.network.encog;

import be.hogent.captchasolvingnetwork.network.encog.util.PropagationType;
import be.hogent.captchasolvingnetwork.network.NeuralNetwork;
import static be.hogent.captchasolvingnetwork.network.encog.util.
    PropagationType.ManhattanPropagation;
import java.util.List;
import org.apache.log4j.Logger;
import org.encog.engine.network.activation.ActivationSigmoid;
import org.encog.ml.data.MLDataSet;
import org.encog.ml.data.basic.BasicMLDataSet;
import org.encog.ml.train.MLTrain;
import org.encog.ml.train.strategy.Strategy;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.training.propagation.back.Backpropagation;
import org.encog.neural.networks.training.propagation.manhattan.
    ManhattanPropagation;
```

```
import org.encog.neural.networks.training.propagation.resilient.
    ResilientPropagation;
import org.encog.neural.networks.training.propagation.scg.
    ScaledConjugateGradient;
import org.encog.util.simple.EncogUtility;

/**
 * EncogBasicNetwork.java (UTF-8)
 *
 * Provides a configurable Encog BasicNetwork
 *
 * 2013/05/19
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.1.0
 */
public class EncogBasicNetwork extends NeuralNetwork {

    private static final Logger logger;
    private double trainingInput[][];
    private double trainingIdeal[][];
    private BasicNetwork network;
    private int[] hiddenLayers;
    private double accuracy;
    private double learningRate;
    private List<Strategy> trainingStrategies;
    private PropagationType propagationType;

    static {
        logger = Logger.getLogger(EncogBasicNetwork.class);
    }

    /**
     * Constructor
     *
     * @param id the id of the network
     * @param trainingInput The inputs for the training
     * @param trainingIdeal the expected results for the training
     * @param hiddenLayers the amount of neuron each hidden layer has (in
         order)
     * @param acuracy the desired accuracy
     * @param learningRate the learning rate (only used with
     * ManhattanPropagation)
     * @param trainingStrategies the training strategies to be used
     */
    protected EncogBasicNetwork(int id, int hSize, int vSize, double[][]
        trainingInput, double[][] trainingIdeal, int[] hiddenLayers, double
        accuracy, double learningRate, List<Strategy> trainingStrategies,
        PropagationType propagationType) {
        super(id, hSize, vSize);
        this.trainingInput = trainingInput;
        this.trainingIdeal = trainingIdeal;
        this.hiddenLayers = hiddenLayers;
        this.accuracy = accuracy;
        this.learningRate = learningRate;
        this.trainingStrategies = trainingStrategies;
        this.propagationType = propagationType;
    }
    @Override
    public void buildNetwork() {
        System.out.println("Building␣basic␣network");
        this.network = new BasicNetwork();

        System.out.println("Adding␣layers␣to␣network");
        network.addLayer(new BasicLayer(null, true, (super.getHsize() *
            super.getVsize())));
        if (hiddenLayers != null) {
            for (int i : hiddenLayers) {
                network.addLayer(new BasicLayer(new ActivationSigmoid(),
                    true, i));
            }
        }
        network.addLayer(new BasicLayer(new ActivationSigmoid(), true,
            trainingIdeal[0].length));

        network.getStructure().finalizeStructure();
        network.reset();
    }

    @Override
    public void trainNetwork() {
        network.reset();

        System.out.println("initializing␣network␣training␣system");
        MLDataSet trainingSet = new BasicMLDataSet(trainingInput,
            trainingIdeal);
        final MLTrain training;

        switch (propagationType) {
            case Backpropagation:
                training = new Backpropagation(network, trainingSet);
                break;
            case ManhattanPropagation:
                training = new ManhattanPropagation(network, trainingSet,
                    learningRate);
                break;
            case ResilientPropagation:
                training = new ResilientPropagation(network, trainingSet);
                break;
            case ScaledConjugateGradient:
                training = new ScaledConjugateGradient(network, trainingSet)
                    ;
                break;
            default:
                IllegalArgumentException e = new IllegalArgumentException("
                    Unknown␣propagationType");
                logger.error("Error␣in␣training␣network.␣Unknow␣propagation␣
                    type", e);
                throw e;
        }

        System.out.println("Propagation:␣" + propagationType.name());

        System.out.println("adding␣training␣strategies");

        for (Strategy strategy : trainingStrategies) {
            training.addStrategy(strategy);
        }

        System.out.println("Start␣training␣to␣acuracy:␣" + accuracy);
        int layers = network.getLayerCount();
        System.out.println("#Layer: " + layers);
        for (int i = 0; i < layers; i++) {
            System.out.println("Layer " + i + " #neurons: " + network.
                getLayerTotalNeuronCount(i));
        }

        long startTimeLong = System.nanoTime();
        EncogUtility.trainToError(training, accuracy);
        long endTimeLong = System.nanoTime();
        double durationInSec = (double) ((endTimeLong - startTimeLong) /
            Math.pow(10, 9));
        System.out.println("Finished training network in: " + durationInSec)
            ;
    }

    @Override
    public double[] evaluate(double[] input, int maxIterations) {
        double[] output = new double[trainingIdeal[0].length];
        System.out.println("Evaluating input");
        long startTimeLong = System.nanoTime();
        network.compute(input, output);
        long endTimeLong = System.nanoTime();
        double durationInSec = (double) ((endTimeLong - startTimeLong) /
            Math.pow(10, 9));
        System.out.println("Finished evaluating in: " + durationInSec);

        return output;
    }

    @Override
    public String getLayerLayout() {
        StringBuilder strBuilder = new StringBuilder();
        strBuilder.append("[ ");
        int layers = network.getLayerCount();
        for (int i = 0; i < layers; i++) {
            strBuilder.append(network.getLayerTotalNeuronCount(i) - 1).
                append(" ");
        }

        return strBuilder.append("]").toString();
    }
}
```

Listing A.41: be.hogent.captchasolvingnetwork.network.encog.EncogBasicNetworkBuilder

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchasolvingnetwork.network.encog;

import be.hogent.captchasolvingnetwork.network.encog.util.PropagationType;
import java.util.ArrayList;
import java.util.List;
import org.encog.ml.train.strategy.Strategy;

/**
 * EncogBasicNetworkBuilder.java (UTF-8)
 *
 * Provides a builder for a configurable Encog BasicNetwork
 *
 * 2013/05/19
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.1.0
 */
public class EncogBasicNetworkBuilder {
    private int id;
    private double[][] trainingInput;
    private double[][] trainingIdeal;
    private int[] hiddenLayers;
    private double accuracy;
    private double learningRate;
    private List<Strategy> trainingStrategies;
    private PropagationType propagationType;
    private int hSize;
    private int vSize;

    /**
     * builderConstructor
     *
     * @param trainingInput The inputs for the training
     * @param trainingIdeal the expected results for the training
     */
    public EncogBasicNetworkBuilder(double[][] trainingInput, double[][]
        trainingIdeal) {
        this.id = -1;
        this.accuracy = 0.00000000001;
        this.learningRate = 2;
        this.trainingStrategies = new ArrayList<>();
        this.propagationType = PropagationType.ResilientPropagation;
        this.trainingInput = trainingInput;
        this.trainingIdeal = trainingIdeal;
        this.hSize = 40;
        this.vSize = 50;
    }

    public EncogBasicNetworkBuilder setId(int id) {
        this.id = id;
        return this;
    }

    public EncogBasicNetworkBuilder setHsize(int hSize) {
        this.hSize = hSize;
        return this;
    }

    public EncogBasicNetworkBuilder setVsize(int vSize) {
        this.vSize = vSize;
        return this;
    }

    public EncogBasicNetworkBuilder setHiddenLayers(int[] hiddenLayers) {
        this.hiddenLayers = hiddenLayers;
        return this;
    }

    public EncogBasicNetworkBuilder setAccuracy(double accuracy) {
        this.accuracy = accuracy;
        return this;
    }

    public EncogBasicNetworkBuilder setLearningRate(double learningRate) {
        this.learningRate = learningRate;
        return this;
    }

    public EncogBasicNetworkBuilder setTrainingStrategies(List<Strategy>
        trainingStrategies) {
        this.trainingStrategies = trainingStrategies;
        return this;
    }

    public EncogBasicNetworkBuilder setPropagationType(PropagationType
        propagationType) {
        this.propagationType = propagationType;
        return this;
    }

    public EncogBasicNetwork createEncogBasicLetterRecognitionNetwork() {
        return new EncogBasicNetwork(id, hSize, vSize, trainingInput,
            trainingIdeal, hiddenLayers, accuracy, learningRate,
            trainingStrategies, propagationType);
    }

}
```

Listing A.42: be.hogent.captchasolvingnetwork.network.encog.EncogHopfieldNetwork

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchasolvingnetwork.network.encog;

import be.hogent.captchasolvingnetwork.network.NeuralNetwork;
import org.apache.log4j.Logger;
import org.encog.ml.data.specific.BiPolarNeuralData;
import org.encog.neural.thermal.HopfieldNetwork;

/**
 * EncogBasicNetwork.java (UTF-8)
 *
 * Provides a configurable Encog HopfieldNetwork
 *
 * 2013/05/19
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class EncogHopfieldNetwork extends NeuralNetwork {

    private static final Logger logger;
    private double trainingInput[][];
    private HopfieldNetwork network;
    private final int neuroncount;

    static {
        logger = Logger.getLogger(EncogBasicNetwork.class);
    }

    /**
     * Constructor
     *
     * @param trainingInput the inputs for training the network
     * @param id the network id
     * @param hSize the horizontal size of the network
     * @param vSize the vertical size of the network
     */
    protected EncogHopfieldNetwork(double[][] trainingInput, int id, int
        hSize, int vSize) {
        super(id, hSize, vSize);
        this.trainingInput = trainingInput;
        neuroncount = vSize*hSize;

        if (neuroncount != trainingInput[0].length) {
            IllegalArgumentException e = new IllegalArgumentException("the␣
                length␣of␣the␣trainingsinputs␣and␣the␣neuroncount␣do␣not␣
                match");
            logger.error(e.getMessage(), e);
            throw e;
        }
    }

    @Override
    public void buildNetwork() {
        System.out.println("Building␣hopfield␣network");
        network = new HopfieldNetwork(neuroncount);
    }

    @Override
    public void trainNetwork() {
        network.reset();
        System.out.println("Training␣hopfield␣network");
        long startTimeLong = System.nanoTime();
        for (double[] ds : trainingInput) {
            network.addPattern(doubleArrayToBiPolarNeuralData(ds));
        }
        long endTimeLong = System.nanoTime();
        double durationInSec = (double) ((endTimeLong - startTimeLong) /
            Math.pow(10, 9));
        System.out.println("Finished␣training␣network␣in:␣" + durationInSec)
            ;
    }

    private BiPolarNeuralData doubleArrayToBiPolarNeuralData(double[] data)
        {
        BiPolarNeuralData patternData = new BiPolarNeuralData(neuroncount);
        if (data.length != neuroncount) {
            IndexOutOfBoundsException e = new IndexOutOfBoundsException("the
                ␣size␣of␣the␣traingsinputs␣is␣different␣from␣the␣amount␣of␣
                input␣neurons");
            logger.error(e.getMessage(), e);
            throw e;
        }
        patternData.setData(data);
        return patternData;
    }

    @Override
    public double[] evaluate(double[] input, int maxIterations) {
        System.out.println("hopfield␣network␣evaluating␣with␣max␣iterations:
            ␣" + maxIterations);
        BiPolarNeuralData inputPattern = doubleArrayToBiPolarNeuralData(
            input);
        network.setCurrentState(inputPattern);
        int cycles = network.runUntilStable(maxIterations);
        System.out.println("Cycles␣until␣stable(max␣" + maxIterations + "):␣
            " + cycles + ",␣result=");
        BiPolarNeuralData outputPattern = (BiPolarNeuralData) network.
            getCurrentState();
        System.out.println(convertForDisplay(inputPattern, outputPattern));
        return outputPattern.getData();
    }

    private String convertForDisplay(BiPolarNeuralData inputPattern,
        BiPolarNeuralData outputPattern) {
        int index1 = 0;
        int index2 = 0;
        StringBuilder block = new StringBuilder();

        for (int row = 0; row < super.getVsize(); row++) {


            for (int col = 0; col < super.getHsize(); col++) {
                if (inputPattern.getBoolean(index1++)) {
                    block.append('O');
                } else {
                    block.append('␣');
                }
            }

            block.append("␣␣␣->␣␣␣");

            for (int col = 0; col < super.getHsize(); col++) {
                if (outputPattern.getBoolean(index2++)) {
                    block.append('O');
                } else {
                    block.append('␣');
                }
            }

            block.append("\n");
        }

        return block.toString();
    }

    @Override
    public String getLayerLayout() {
        return "[␣" + getHsize() + "␣X␣" + getVsize() + "␣]";
    }
}
```

Listing A.43: be.hogent.captchasolvingnetwork.network.encog.EncogHopfieldNetworkBuilder

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
```

```
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchasolvingnetwork.network.encog;

/**
 * EncogBasicNetworkBuilder.java (UTF-8)
 *
 * Provides a builder for a configurable Encog HopfieldNetwork
 *
 * 2013/05/19
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class EncogHopfieldNetworkBuilder {

    private double[][] trainingInput;
    private int id;
    private int hSize;
    private int vSize;

    public EncogHopfieldNetworkBuilder(double[][] trainingInput, int hSize,
        int vSize) {
        this.trainingInput = trainingInput;
        this.hSize = hSize;
        this.vSize = vSize;
        this.id = -1;
    }

    public EncogHopfieldNetworkBuilder setId(int id) {
        this.id = id;
        return this;
    }

    public EncogHopfieldNetwork createEncogHopfieldNetwork() {
        return new EncogHopfieldNetwork(trainingInput, id, hSize, vSize);
    }
}
```

A.35 Package be.hogent.captchasolvingnetwork.network.en

Listing A.44: be.hogent.bulksolvingstatistics.domain.neuralnetwork.dataobjects.NeuralNetworkDataObject

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.bulksolvingstatistics.domain.neuralnetwork.dataobjects;

/**
 * NeuralNetworkDataObject.java (UTF-8)
 *
 * This class will act as data container for neural network data, this is to
 * prevent using to much memory compared to storing the complete networks.
 *
 * 2013/05/20
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class NeuralNetworkDataObject {

    private int id;
    private String networkType;
    private String layerLayout;
    private double accuracy;
    private double trainingDuration;
    private int iterations;
    private String savedLocation;

    protected NeuralNetworkDataObject(int id, String networkType, String
        layerLayout, double accuracy, double trainingDuration, int
        iterations, String savedLocation) {
        this.id = id;
        this.networkType = networkType;
        this.layerLayout = layerLayout;
        this.accuracy = accuracy;
        this.trainingDuration = trainingDuration;
        this.iterations = iterations;
        this.savedLocation = savedLocation;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNetworkType() {
        return networkType;
    }

    public void setNetworkType(String networkType) {
        this.networkType = networkType;
    }

    public String getLayerLayout() {
        return layerLayout;
    }

    public void setLayerLayout(String layerLayout) {
        this.layerLayout = layerLayout;
    }

    public double getAccuracy() {
        return accuracy;
    }

    public void setAccuracy(double accuracy) {
        this.accuracy = accuracy;
    }

    public double getTrainingDuration() {
        return trainingDuration;
    }

    public void setTrainingDuration(double trainingDuration) {
        this.trainingDuration = trainingDuration;
    }

    public int getIterations() {
        return iterations;
    }

    public void setIterations(int iterations) {
        this.iterations = iterations;
    }

    public String getSavedLocation() {
        return savedLocation;
    }

    public void setSavedLocation(String savedLocation) {
        this.savedLocation = savedLocation;
    }
}
```

Listing A.45: be.hogent.bulksolvingstatistics.domain.neuralnetwork.dataobjects.NeuralNetworkDataObject

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.bulksolvingstatistics.domain.neuralnetwork.dataobjects;

/**
 * NeuralNetworkDataObjectBuilder.java (UTF-8)
 *
 * This class will act as builder for a NeuralNetworkDataObject instance
 *
 * 2013/05/20
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class NeuralNetworkDataObjectBuilder {
    private int id;
    private String networkType;
    private String layerLayout;
    private double accuracy;
    private double trainingDuration;
    private int iterations;
    private String savedLocation;

    public NeuralNetworkDataObjectBuilder() {
    }
    public NeuralNetworkDataObjectBuilder setId(int id) {
        this.id = id;
        return this;
    }

    public NeuralNetworkDataObjectBuilder setNetworkType(String networkType)
         {
        this.networkType = networkType;
        return this;
    }

    public NeuralNetworkDataObjectBuilder setLayerLayout(String layerLayout)
         {
        this.layerLayout = layerLayout;
        return this;
    }

    public NeuralNetworkDataObjectBuilder setAccuracy(double accuracy) {
        this.accuracy = accuracy;
        return this;
    }

    public NeuralNetworkDataObjectBuilder setTrainingDuration(double
        trainingDuration) {
        this.trainingDuration = trainingDuration;
        return this;
    }

    public NeuralNetworkDataObjectBuilder setIterations(int iterations) {
        this.iterations = iterations;
        return this;
    }

    public NeuralNetworkDataObjectBuilder setSavedLocation(String
        savedLocation) {
        this.savedLocation = savedLocation;
        return this;
    }

    public NeuralNetworkDataObject createNeuralNetworkDataObject() {
        return new NeuralNetworkDataObject(id, networkType, layerLayout,
            accuracy, trainingDuration, iterations, savedLocation);
    }

}
```

Listing A.46: be.hogent.bulksolvingstatistics.domain.neuralnetwork.dataobjects.TestResultDataOb

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.bulksolvingstatistics.domain.neuralnetwork.dataobjects;

/**
 * TestResultDataObject.java (UTF-8)
 *
 * This class will act as data container for test result data, this is to
 * prevent using to much memory compared to storing the complete test
     results.
 *
 * 2013/05/20
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class TestResultDataObject {

    private int id;
    private int networkID;
    private String testType;
    private double duration;
    private String character;
    private boolean correct;

    protected TestResultDataObject(int id, int networkID, String testType,
        double duration, String character, boolean correct) {
        this.id = id;
        this.networkID = networkID;
        this.testType = testType;
        this.duration = duration;
        this.character = character;
        this.correct = correct;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getNetworkID() {
        return networkID;
    }

    public void setNetworkID(int networkID) {
        this.networkID = networkID;
    }

    public String getTestType() {
        return testType;
    }

    public void setTestType(String testType) {
        this.testType = testType;
    }

    public double getDuration() {
        return duration;
    }

    public void setDuration(double duration) {
        this.duration = duration;
    }

    public String getCharacter() {
        return character;
    }

    public void setCharacter(String character) {
        this.character = character;
    }

    public boolean isCorrect() {
        return correct;
    }

    public void setCorrect(boolean correct) {
        this.correct = correct;
    }
}
```

Listing A.47: be.hogent.bulksolvingstatistics.domain.neuralnetwork.dataobjects.TestResultDataOb

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.bulksolvingstatistics.domain.neuralnetwork.dataobjects;

/**
 * TestResultDataObjectBuilder.java (UTF-8)
 *
 * This class will act as builder for a TestResultDataObjectBuilder instance
 *
 * 2013/05/20
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class TestResultDataObjectBuilder {
    private int id;
    private int networkID;
    private String testType;
    private double duration;
    private String character;
    private boolean correct;

    public TestResultDataObjectBuilder() {
    }

    public TestResultDataObjectBuilder setID(int id) {
        this.id = id;
        return this;
    }

    public TestResultDataObjectBuilder setNetworkID(int networkID) {
        this.networkID = networkID;
        return this;
    }

    public TestResultDataObjectBuilder setTestType(String testType) {
        this.testType = testType;
        return this;
    }

    public TestResultDataObjectBuilder setDuration(double duration) {
        this.duration = duration;
        return this;
    }

    public TestResultDataObjectBuilder setCharacter(String character) {
        this.character = character;
        return this;
    }
    public TestResultDataObjectBuilder setCorrect(boolean correct) {
        this.correct = correct;
        return this;
    }

    public TestResultDataObject createTestResultDataObject() {
        return new TestResultDataObject(id, networkID, testType, duration,
            character, correct);
    }

}
```

Listing A.48: be.hogent.bulksolvingstatistics.domain.neuralnetwork.encogutils.EncogTrainingSet

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package be.hogent.bulksolvingstatistics.domain.neuralnetwork.encogutils;

import be.hogent.captchabuilder.elementcreator.renderer.text.
    AbstractWordRenderer;
import be.hogent.captchabuilder.elementcreator.renderer.text.
    DefaultWordRenderer;
import be.hogent.captchabuilder.elementcreator.renderer.text.WordRenderer;
import be.hogent.captchabuilder.util.ColorRangeRGBA;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import be.hogent.captchasolvingnetwork.util.CharacterPatternUtils;
import be.hogent.captchasolvingnetwork.util.ImageToInputPattern;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.geom.AffineTransform;
import java.awt.image.AffineTransformOp;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;

/**
 * EncogTrainingSet.java (UTF-8)
 *
 * Utility class to help generate the input and output trainingsets for an
 *   encog
 * Neural Network.
 *
 * 2013/05/20
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public class EncogTrainingSet {

    public static double[][] buildTrainingInputSet(char[] chars, int hSize,
        int vSize) {
        double[][] inputTrainingsSet = new double[chars.length][];
        System.out.println("building␣Trainingsets");
        BufferedImage img;
        WordRenderer renderer = new DefaultWordRenderer(new ColorRangeRGBA
            (0, 0, 0, 255), AbstractWordRenderer.DEFAULT_FONTS, 0, 0.25,
            CaptchaConstants.DEFAULT_STROKE_WIDTH);
        int index = 0;

        for (char c : chars) {
            img = new BufferedImage(40, 50, BufferedImage.TYPE_INT_ARGB);
            renderer.render(String.valueOf(c), img);

            // check if size == the default size (40*50) if not scale
            if (hSize != 40 || vSize != 50) {
                BufferedImage resized = new BufferedImage(hSize, vSize, img.
                    getType());
                Graphics2D g = resized.createGraphics();
                g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
                    RenderingHints.VALUE_INTERPOLATION_BILINEAR);
                g.drawImage(img, 0, 0, hSize, vSize, 0, 0, img.getWidth(),
                    img.getHeight(), null);
                g.dispose();

                //replace the origal with the resized
                img = resized;
            }

            try {
                String path = "TrainingsetImages/";
                // if the directory does not exist, create it and it's
                    parents
                File theDir = new File(path);
                if (!theDir.exists()) {
                    System.out.println("creating directory: " + path);
                    boolean result = theDir.mkdirs();
                    if (result) {
                        System.out.println("Directory created");
                    }
                }

                ImageIO.write(img, "png", new File(path + Character.getName(
                    c) + "-" + hSize + "X" + vSize + ".png"));
            } catch (IOException ex) {
                System.err.println(ex.getMessage());
            }

            inputTrainingsSet[index++] = ImageToInputPattern.
                colorRangeToDoubleInputPattern(img, 0, 0);
        }

        return inputTrainingsSet;
    }

    public static double[][] buildTrainingIdealSet(char[] chars) {
        double[][] outputTrainingsSet = new double[chars.length][];
        System.out.println("building TrainingIdealSet");
        int index = 0;

        for (char c : chars) {
            outputTrainingsSet[index++] = CharacterPatternUtils.
                characterToBitArray(c);
        }

        return outputTrainingsSet;
    }
```

```
}
```

Listing A.49: be.hogent.captchabuilder.elementcreator.producer.background.AbstractBackgroundI

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.background;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import java.awt.image.BufferedImage;

/**
 * AbstractBackgroundProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.4
 * @version 1.1.0
 */
public abstract class AbstractBackgroundProducer implements
    BackgroundProducer {

    protected ColorRangeRGBA  colorRange1;
    protected ColorRangeRGBA  colorRange2;

    protected AbstractBackgroundProducer(ColorRangeRGBA  colors1Range,
        ColorRangeRGBA  colors2Range) {
        this.colorRange1 = colors1Range;
        this.colorRange2 = colors2Range;
    }

    @Override
    public BufferedImage addBackground(BufferedImage image) {
        return getBackground(image.getWidth(), image.getHeight());
    }
}
```

Listing A.50: be.hogent.captchabuilder.elementcreator.producer.background.BackgroundProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.background;

import java.awt.image.BufferedImage;

/**
 * BackgroundProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.4
 * @version 1.0.7
 */
public interface BackgroundProducer {

    /**
```

```
     * Add the background to the given image.
     *
     * @param image The image onto which the background will be rendered.
     * @return The image with the background rendered.
     */
    public BufferedImage addBackground(BufferedImage image);

    public BufferedImage getBackground(int width, int height);
}
```

Listing A.51: be.hogent.captchabuilder.elementcreator.producer.background.BackgroundProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.background;

import be.hogent.captchabuilder.elementcreator.CaptchaElementCreatorBuilder;
import be.hogent.captchabuilder.util.ColorRangeRGBA;
import be.hogent.captchabuilder.util.enums.producer.BackgroundProducerType;
import static be.hogent.captchabuilder.util.enums.producer.
    BackgroundProducerType.FLATCOLOR;
import static be.hogent.captchabuilder.util.enums.producer.
    BackgroundProducerType.SQUIGGLES;
import static be.hogent.captchabuilder.util.enums.producer.
    BackgroundProducerType.TRANSPARENT;
import static be.hogent.captchabuilder.util.enums.producer.
    BackgroundProducerType.TWOCOLORGRADIENT;

/**
 * BackgroundProducerBuilder.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.4
 * @version 1.1.0
 */
public class BackgroundProducerBuilder implements
    CaptchaElementCreatorBuilder {

    private ColorRangeRGBA colorRange1;
    private ColorRangeRGBA colorRange2;
    private BackgroundProducerType type;

    public BackgroundProducerBuilder(BackgroundProducerType type) {
        this.type = type;

        switch (type) {
            case FLATCOLOR:
                colorRange1 = new ColorRangeRGBA(222, 222, 222);
                colorRange2 = new ColorRangeRGBA(222, 222, 222);
                break;
            case SQUIGGLES:
                colorRange1 = new ColorRangeRGBA(0);
                colorRange2 = new ColorRangeRGBA(0);
                break;
            case TRANSPARENT:
                colorRange1 = new ColorRangeRGBA(255, 255, 255);
                colorRange2 = new ColorRangeRGBA(255, 255, 255);
                break;
            case TWOCOLORGRADIENT:
                colorRange1 = new ColorRangeRGBA(0, 0, 255);
                colorRange2 = new ColorRangeRGBA(0, 255, 0);
                break;
            default:
                colorRange1 = new ColorRangeRGBA(211, 211, 211);
                colorRange2 = new ColorRangeRGBA(169, 169, 169);
        }
    }

    public BackgroundProducerBuilder setColorRange1(ColorRangeRGBA
        colorRange1) {
        this.colorRange1 = colorRange1;
        return this;
    }

    public BackgroundProducerBuilder setColorRange2(ColorRangeRGBA
        colorRange2) {
        this.colorRange2 = colorRange2;
        return this;
    }

    @Override
    public BackgroundProducer create() {
        switch (type) {
            case FLATCOLOR:
                return new FlatColorBackgroundProducer(colorRange1,
                    colorRange2);
            case SQUIGGLES:
                return new SquigglesBackgroundProducer(colorRange1,
                    colorRange2);
            case TRANSPARENT:
                return new TransparentBackgroundProducer(colorRange1,
                    colorRange2);
            case TWOCOLORGRADIENT:
                return new TwoColorGradientBackgroundProducer(colorRange1,
                    colorRange2);
            default:
                throw new IllegalArgumentException("Background␣producer␣not␣
                    found:␣" + type.name());
        }
    }
}
```

Listing A.52: be.hogent.captchabuilder.elementcreator.producer.background.FlatColorBackground

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.background;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import java.awt.Graphics2D;
import java.awt.geom.Rectangle2D;
import java.awt.image.BufferedImage;

/**
 * FlatColorBackgroundProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.4
 * @version 1.1.0
 */
public class FlatColorBackgroundProducer extends AbstractBackgroundProducer
    {

    protected FlatColorBackgroundProducer(ColorRangeRGBA colorRange1,
        ColorRangeRGBA colorRange2) {
        super(colorRange1, colorRange2);
    }

    @Override
    public BufferedImage getBackground(int width, int height) {
        BufferedImage img = new BufferedImage(width, height,
                BufferedImage.TYPE_INT_RGB);

        Graphics2D graphics = img.createGraphics();
        graphics.setPaint(colorRange1.getRandomColorInRange());
        graphics.fill(new Rectangle2D.Double(0, 0, width, height));
        graphics.drawImage(img, 0, 0, null);
        graphics.dispose();

        return img;
    }
}
```

Listing A.53: be.hogent.captchabuilder.elementcreator.producer.background.SquigglesBackgroundProduce

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.background;
```

```
import be.hogent.captchabuilder.util.ColorRangeRGBA;
import java.awt.AlphaComposite;
import java.awt.BasicStroke;
import java.awt.Graphics2D;
import java.awt.geom.Arc2D;
import java.awt.image.BufferedImage;

/**
 * SquigglesBackgroundProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.4
 * @version 1.1.0
 */
public class SquigglesBackgroundProducer extends AbstractBackgroundProducer
    {

    protected SquigglesBackgroundProducer(ColorRangeRGBA colorRange1,
        ColorRangeRGBA colorRange2) {
        super(colorRange1, colorRange2);
    }

    @Override
    public BufferedImage getBackground(int width, int height) {
        BufferedImage result = new BufferedImage(width, height,
                BufferedImage.TYPE_INT_RGB);
        Graphics2D graphics = result.createGraphics();

        BasicStroke bs = new BasicStroke(2.0f, BasicStroke.CAP_BUTT,
            BasicStroke.JOIN_MITER, 2.0f, new float[]{2.0f, 2.0f}, 0.0f);
        graphics.setStroke(bs);
        AlphaComposite ac = AlphaComposite.getInstance(AlphaComposite.
            SRC_OVER,
                0.75f);
        graphics.setComposite(ac);

        graphics.translate(width * -1.0, 0.0);
        double delta = 15.0;
        double xt;
        for (xt = 0.0; xt < (2.0 * width); xt += delta) {
            Arc2D arc = new Arc2D.Double(0, 0, width, height, 0.0, 360.0,
                Arc2D.OPEN);
            graphics.draw(arc);
            graphics.translate(delta, 0.0);
        }
        graphics.dispose();
        return result;
    }
}
```

Listing A.54: be.hogent.captchabuilder.elementcreator.producer.background.TransparentBackgrou

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.background;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import java.awt.AlphaComposite;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;

/**
 * TransparentBackgroundProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.4
 * @version 1.1.0
 */
public class TransparentBackgroundProducer extends
    AbstractBackgroundProducer {

    protected TransparentBackgroundProducer(ColorRangeRGBA colorRange1,
        ColorRangeRGBA colorRange2) {
        super(colorRange1, colorRange2);
    }

    @Override
    public BufferedImage getBackground(int width, int height) {
        BufferedImage bg = new BufferedImage(width, height, BufferedImage.
            TRANSLUCENT);
        Graphics2D g = bg.createGraphics();
        g.setComposite(AlphaComposite.getInstance(AlphaComposite.CLEAR, 0.0f
            ));
        g.fillRect(0, 0, width, height);

        return bg;
    }
}
```

Listing A.55: be.hogent.captchabuilder.elementcreator.producer.background.TwoColorGradientBa

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.background;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import java.awt.GradientPaint;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.geom.Rectangle2D;
import java.awt.image.BufferedImage;

/**
 * TwoColorGradientBackgroundProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.4
 * @version 1.1.0
 */
public class TwoColorGradientBackgroundProducer extends
    AbstractBackgroundProducer {

    protected TwoColorGradientBackgroundProducer(ColorRangeRGBA colorRange1,
         ColorRangeRGBA colorRange2) {
        super(colorRange1, colorRange2);
    }

    @Override
    public BufferedImage getBackground(int width, int height) {
        // create an opaque image
        BufferedImage img = new BufferedImage(width, height,
                BufferedImage.TYPE_INT_RGB);

        Graphics2D g = img.createGraphics();
        RenderingHints hints = new RenderingHints(
                RenderingHints.KEY_ANTIALIASING,
                RenderingHints.VALUE_ANTIALIAS_ON);

        g.setRenderingHints(hints);

        // create the gradient color
        GradientPaint ytow = new GradientPaint(0, 0, colorRange1.
            getRandomColorInRange(), width, height,
                colorRange2.getRandomColorInRange());

        g.setPaint(ytow);
        // draw gradient color
        g.fill(new Rectangle2D.Double(0, 0, width, height));

        // draw the transparent image over the background
        g.drawImage(img, 0, 0, null);
        g.dispose();

        return img;
    }

}
```

Listing A.56: be.hogent.captchabuilder.elementcreator.producer.border.AbstractBorderProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.border;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import java.awt.AlphaComposite;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;

/**
 * AbstractBorderProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/18
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.12
 * @version 1.1.0
 */
public abstract class AbstractBorderProducer implements BorderProducer {

    protected ColorRangeRGBA  colorRange;
    protected int thickness;

    protected AbstractBorderProducer(ColorRangeRGBA  colorRange, int
        thickness) {
        this.colorRange = colorRange;
        this.thickness = thickness;
    }

    @Override
    public void addBorder(BufferedImage img) {
        int width = img.getWidth();
        int height = img.getHeight();
        Graphics2D g = img.createGraphics();
        g.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER,
            1.0f));
        g.setColor(colorRange.getRandomColorInRange());
        setStrokeOptions(g);
        g.drawLine(0, 0, 0, width);
        g.drawLine(0, 0, width, 0);
        g.drawLine(0, height, width, height);
        g.drawLine(width, height, width, 0);
    }
}
```

Listing A.57: be.hogent.captchabuilder.elementcreator.producer.border.BorderProducer

```
/*
```

```
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.border;

import java.awt.Graphics2D;
import java.awt.image.BufferedImage;

/**
 * BorderProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/18
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.12
 * @version 1.0.12
 */
public interface BorderProducer {

    public void addBorder(BufferedImage img);

     public void setStrokeOptions(Graphics2D g);

}
```

Listing A.58: be.hogent.captchabuilder.elementcreator.producer.border.BorderProducerBuilder

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.border;

import be.hogent.captchabuilder.elementcreator.CaptchaElementCreatorBuilder;
import be.hogent.captchabuilder.util.ColorRangeRGBA;
import be.hogent.captchabuilder.util.enums.producer.BorderProducerType;

/**
 * BorderProducerBuilder.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/12
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.12
 * @version 1.0.12
 */
public class BorderProducerBuilder implements CaptchaElementCreatorBuilder {

    private ColorRangeRGBA colorRange;
    private int thickness;
    private BorderProducerType type;

    public BorderProducerBuilder(BorderProducerType type) {
        this.type = type;
        this.colorRange = new ColorRangeRGBA(0);
        this.thickness = 1;
    }

    public BorderProducerBuilder setColorRange(ColorRangeRGBA colorRange) {
        this.colorRange = colorRange;
        return this;
    }
    public BorderProducerBuilder setThickness(int thickness) {
        this.thickness = thickness;
        return this;
    }

    @Override
    public BorderProducer create() {
        switch (type) {
            case SOLID:
                return new SolidBorderProducer(colorRange, thickness);
            default:
                throw new IllegalArgumentException("Border␣producer␣not␣
                    found:␣" + type.name());
        }
    }
}
```

Listing A.59: be.hogent.captchabuilder.elementcreator.producer.border.SolidBorderProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.border;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import java.awt.BasicStroke;
import java.awt.Graphics2D;

/**
 * SolidBorderProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/18
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.12
 * @version 1.1.0
 */
public class SolidBorderProducer extends AbstractBorderProducer {

    public SolidBorderProducer(ColorRangeRGBA colorRange, int thickness) {
        super(colorRange, thickness);
    }

    @Override
    public void setStrokeOptions(Graphics2D g) {
        g.setStroke(new BasicStroke(thickness));
    }

}
```

Listing A.60: be.hogent.captchabuilder.elementcreator.producer.noise.AbstractNoiseProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.noise;

import be.hogent.captchabuilder.util.ColorRangeRGBA;

/**
 * AbstractNoiseProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.5
 * @version 1.1.0
 */
public abstract class AbstractNoiseProducer implements NoiseProducer {

    protected float thickness;
    protected ColorRangeRGBA colorRange;

    protected AbstractNoiseProducer(float thickness, ColorRangeRGBA
        colorRange) {
        this.thickness = thickness;
        this.colorRange = colorRange;
    }
}
```

Listing A.61: be.hogent.captchabuilder.elementcreator.producer.noise.CurvedLineNoiseProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.noise;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import java.awt.BasicStroke;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.geom.CubicCurve2D;
import java.awt.geom.PathIterator;
import java.awt.geom.Point2D;
import java.awt.image.BufferedImage;
import java.util.Random;

/**
 * CurvedLineNoiseProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.5
 * @version 1.1.0
 */
public class CurvedLineNoiseProducer extends AbstractNoiseProducer {

    protected CurvedLineNoiseProducer(float thickness, ColorRangeRGBA
        colorRange) {
        super(thickness, colorRange);
    }

    @Override
    public void makeNoise(BufferedImage image) {
        Random RAND = CaptchaConstants.RANDOM;
        int width = image.getWidth();
        int height = image.getHeight();

        // the curve from where the points are taken
        CubicCurve2D cc = new CubicCurve2D.Float(width * .1f, height
                * RAND.nextFloat(), width * .1f, height
                * RAND.nextFloat(), width * .25f, height
                * RAND.nextFloat(), width * .9f, height
                * RAND.nextFloat());

        // creates an iterator to define the boundary of the flattened curve
        PathIterator pi = cc.getPathIterator(null, 2);
        Point2D tmp[] = new Point2D[200];
        int i = 0;

        // while pi is iterating the curve, adds points to tmp array
        while (!pi.isDone()) {
            float[] coords = new float[6];
            switch (pi.currentSegment(coords)) {
                case PathIterator.SEG_MOVETO:
                case PathIterator.SEG_LINETO:
                    tmp[i] = new Point2D.Float(coords[0], coords[1]);
            }
            i++;
            pi.next();
        }

        // the points where the line changes the stroke and direction
        Point2D[] pts = new Point2D[i];
        // copies points from tmp to pts
        System.arraycopy(tmp, 0, pts, 0, i);

        Graphics2D graph = (Graphics2D) image.getGraphics();
        graph.setRenderingHints(new RenderingHints(
                RenderingHints.KEY_ANTIALIASING,
                RenderingHints.VALUE_ANTIALIAS_ON));

        graph.setColor(colorRange.getRandomColorInRange());

        // for the maximum 3 point change the stroke and direction
        for (i = 0; i < pts.length - 1; i++) {
            if (i < 3) {
                graph.setStroke(new BasicStroke(thickness));
            }
            graph.drawLine((int) pts[i].getX(), (int) pts[i].getY(),
                    (int) pts[i + 1].getX(), (int) pts[i + 1].getY());
        }

        graph.dispose();
    }
}
```

Listing A.62: be.hogent.captchabuilder.elementcreator.producer.noise.NoiseProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.noise;

import java.awt.image.BufferedImage;

/**
 * NoiseProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.5
 * @version 1.0.7
 */
public interface NoiseProducer {

    public void makeNoise(BufferedImage image);
}
```

Listing A.63: be.hogent.captchabuilder.elementcreator.producer.noise.NoiseProducerBuilder

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.noise;

import be.hogent.captchabuilder.elementcreator.CaptchaElementCreatorBuilder;
import be.hogent.captchabuilder.util.ColorRangeRGBA;
import be.hogent.captchabuilder.util.enums.producer.NoiseProducerType;

/**
 * NoiseProducerBuilder.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.5
 * @version 1.1.0
 */
```

```
public class NoiseProducerBuilder implements CaptchaElementCreatorBuilder {

    private float thickness;
    private ColorRangeRGBA colorRange;
    private NoiseProducerType type;

    public NoiseProducerBuilder(NoiseProducerType type) {
        this.colorRange = new ColorRangeRGBA(0);
        this.type = type;
        this.thickness = 3.5f;
    }

    public NoiseProducerBuilder setThickness(float thickness) {
        this.thickness = thickness;
        return this;
    }

    public NoiseProducerBuilder setColorRange(ColorRangeRGBA colorRange) {
        this.colorRange = colorRange;
        return this;
    }

    @Override
    public NoiseProducer create() {
        switch (type) {
            case CURVEDLINE:
                return new CurvedLineNoiseProducer(thickness, colorRange);
            case STRAIGHTLINE:
                return new StraightLineNoiseProducer(thickness, colorRange);
            default:
                throw new IllegalArgumentException("NoiseProduder␣not␣found:
                    ␣" + type.name());
        }
    }
}
```

Listing A.64: be.hogent.captchabuilder.elementcreator.producer.noise.StraightLineNoiseProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.noise;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;

/**
 * StraightLineNoiseProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.5
 * @version 1.1.0
 */
public class StraightLineNoiseProducer extends AbstractNoiseProducer {

    public StraightLineNoiseProducer(float thickness, ColorRangeRGBA
        colorRange) {
        super(thickness, colorRange);
    }

    @Override
    public void makeNoise(BufferedImage image) {
        Graphics2D graphics = image.createGraphics();
        int height = image.getHeight();
        int width = image.getWidth();
        int y1 = CaptchaConstants.RANDOM.nextInt(height) + 1;
        int y2 = CaptchaConstants.RANDOM.nextInt(height) + 1;
        drawLine(graphics, y1, width, y2);
    }

    private void drawLine(Graphics g, int y1, int x2, int y2) {
        int X1 = 0;

        // The thick line is in fact a filled polygon
        g.setColor(colorRange.getRandomColorInRange());
        int dX = x2 - X1;
        int dY = y2 - y1;
        // line length
        double lineLength = Math.sqrt(dX * dX + dY * dY);

        double scale = thickness / (2 * lineLength);

        // The x and y increments from an endpoint needed to create a
        // rectangle...
        double ddx = -scale * dY;
        double ddy = scale * dX;
        ddx += (ddx > 0) ? 0.5 : -0.5;
        ddy += (ddy > 0) ? 0.5 : -0.5;
        int dx = (int) ddx;
        int dy = (int) ddy;

        // Now we can compute the corner points...
        int xPoints[] = new int[4];
        int yPoints[] = new int[4];

        xPoints[0] = X1 + dx;
        yPoints[0] = y1 + dy;
        xPoints[1] = X1 - dx;
        yPoints[1] = y1 - dy;
        xPoints[2] = x2 - dx;
        yPoints[2] = y2 - dy;
        xPoints[3] = x2 + dx;
        yPoints[3] = y2 + dy;

        g.fillPolygon(xPoints, yPoints, 4);
    }
}
```

Listing A.65: be.hogent.captchabuilder.elementcreator.producer.text.AbstractTextProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.text;

import be.hogent.captchabuilder.util.ArrayUtil;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;

/**
 * AbstractTextProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/14
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.2
 * @version 1.0.7
 */
public abstract class AbstractTextProducer extends ArrayUtil<Character>
    implements TextProducer {

    private final char[] _srcChars;
    private int _minLength;
    private int _maxLength;

    protected AbstractTextProducer(char[] chars, int minLenght, int
        maxLenght) {
        _minLength = minLenght;
        _maxLength = maxLenght;
        _srcChars = chars;
    }

    @Override
    public String getText() {
        String capText = "";
        int _length = Math.max(_minLength, CaptchaConstants.RANDOM.nextInt(
            _maxLength));
        for (int i = 0; i < _length; i++) {
            capText += _srcChars[CaptchaConstants.RANDOM.nextInt(_srcChars.
                length)];
        }

        return capText;
    }

    /*
     * No Longer used
     *
     * private static char[] copyOf(char[] original, int newLength) {
     *    char[] copy = new char[newLength];
     *    System.arraycopy(original, 0, copy, 0,
     *            Math.min(original.length, newLength));
     *    return copy;
     * }
     */

    public void setLength(int minLength, int maxLength) {
        if (minLength < 0 || maxLength < minLength) {
            this._minLength = minLength;
        }
        this._maxLength = maxLength;
    }
}
```

Listing A.66: be.hogent.captchabuilder.elementcreator.producer.text.AlphanumericTextProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.text;

import be.hogent.captchabuilder.util.enums.CaptchaConstants;

/**
 * AlphanumericTextProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/14
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.2
 * @version 1.0.7
 */
public class AlphanumericTextProducer extends AbstractTextProducer {

    protected AlphanumericTextProducer(int minLenght, int maxLenght) {
        super(concat(CaptchaConstants.LETTERS, CaptchaConstants.NUMBERS),
            minLenght, maxLenght);
    }
}
```

Listing A.67: be.hogent.captchabuilder.elementcreator.producer.text.ArabicTextProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.text;

import be.hogent.captchabuilder.util.enums.CaptchaConstants;

/**
 * ArabicTextProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/14
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.2
 * @version 1.0.7
 */
public class ArabicTextProducer extends AbstractTextProducer {

    protected ArabicTextProducer(int minLenght, int maxLenght) {
        // I hope we don't generate something offensive
        super(CaptchaConstants.ARABIC_CHARS, minLenght, maxLenght);
    }
}
```

Listing A.68: be.hogent.captchabuilder.elementcreator.producer.text.ChineseTextProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.text;

/**
 * ChineseTextProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/14
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.2
 * @version 1.0.7
 */
public class ChineseTextProducer extends AbstractTextProducer {

    protected ChineseTextProducer(int minLenght, int maxLenght) {
        super(buildChineseCharset(), minLenght, maxLenght);
    }

    private static char[] buildChineseCharset() {
        // Here's hoping none of the characters in this range are offensive.
        int CODE_POINT_START = 0x4E00;
        int CODE_POINT_END = 0x4F6F;
        int NUM_CHARS = CODE_POINT_END - CODE_POINT_START;
        char[] CHARS;

        CHARS = new char[NUM_CHARS];
        for (char c = (char) CODE_POINT_START, i = 0; c < CODE_POINT_END; c
            ++, i++) {
            CHARS[i] = Character.valueOf(c);
        }

        return CHARS;
    }
}
```

Listing A.69: be.hogent.captchabuilder.elementcreator.producer.text.LetterTextProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.text;

import be.hogent.captchabuilder.util.enums.CaptchaConstants;

/**
 * LetterTextProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/14
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.1
 * @version 1.0.7
 */
public class LetterTextProducer extends AbstractTextProducer {

    protected LetterTextProducer(int minLenght, int maxLenght) {
        super(CaptchaConstants.LETTERS, minLenght, maxLenght);
    }
}
```

Listing A.70: be.hogent.captchabuilder.elementcreator.producer.text.NumbersProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.text;

import be.hogent.captchabuilder.util.enums.CaptchaConstants;

/**
 * NumbersProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/14
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.2
 * @version 1.0.7
 */
public class NumbersProducer extends AbstractTextProducer {

    protected NumbersProducer(int minLenght, int maxLenght) {
        super(CaptchaConstants.NUMBERS, minLenght, maxLenght);
    }
}
```

Listing A.71: be.hogent.captchabuilder.elementcreator.producer.text.ReducedAlphanumericTextProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.text;

import be.hogent.captchabuilder.util.enums.CaptchaConstants;

/**
 * ReducedAlphanumericTextProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/14
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.2
 * @version 1.0.7
 */
public class ReducedAlphanumericTextProducer extends AbstractTextProducer {

    protected ReducedAlphanumericTextProducer(int minLenght, int maxLenght)
        {
        super(CaptchaConstants.REDUCEDALPHANUMERIC, minLenght, maxLenght);
    }
}
```

Listing A.72: be.hogent.captchabuilder.elementcreator.producer.text.SpecialAlphanumericTextPro

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.text;

import be.hogent.captchabuilder.util.enums.CaptchaConstants;

/**
 * SpecialAlphanumericTextProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/14
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.1
 * @version 1.0.7
 */
public class SpecialAlphanumericTextProducer extends AbstractTextProducer {

    protected SpecialAlphanumericTextProducer(int minLenght, int maxLenght)
        {
        super(concat(CaptchaConstants.LETTERS, CaptchaConstants.NUMBERS,
            CaptchaConstants.SPECIAL), minLenght, maxLenght);
    }
}
```

Listing A.73: be.hogent.captchabuilder.elementcreator.producer.text.SpecialLetterTextProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.text;

import be.hogent.captchabuilder.util.enums.CaptchaConstants;

/**
 * SpecialLetterTextProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/14
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.1
 * @version 1.0.7
 */
public class SpecialLetterTextProducer extends AbstractTextProducer {

    protected SpecialLetterTextProducer(int minLenght, int maxLenght) {
        super(concat(CaptchaConstants.LETTERS, CaptchaConstants.SPECIAL),
            minLenght, maxLenght);
    }
}
```

Listing A.74: be.hogent.captchabuilder.elementcreator.producer.text.SpecialNumbersProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.text;

import be.hogent.captchabuilder.util.enums.CaptchaConstants;

/**
 * SpecialNumbersProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/14
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.1
 * @version 1.0.7
 */
public class SpecialNumbersProducer extends AbstractTextProducer {

    protected SpecialNumbersProducer(int minLenght, int maxLenght) {
        super(concat(CaptchaConstants.NUMBERS, CaptchaConstants.SPECIAL),
            minLenght, maxLenght);
    }
}
```

Listing A.75: be.hogent.captchabuilder.elementcreator.producer.text.TextProducer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.text;

/**
 * TextProducer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.4
 * @version 1.0.7
 */
public interface TextProducer {

    public String getText();
}
```

Listing A.76: be.hogent.captchabuilder.elementcreator.producer.text.TextProducerBuilder

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.producer.text;
```

```java
import be.hogent.captchabuilder.elementcreator.CaptchaElementCreatorBuilder;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import be.hogent.captchabuilder.util.enums.producer.TextProducerType;

/**
 * TextProducerBuilder.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.3
 * @version 1.0.13
 */
public class TextProducerBuilder implements CaptchaElementCreatorBuilder {

    private int minLenght;
    private int maxLenght;
    private TextProducerType type;

    public TextProducerBuilder(TextProducerType type) {
        this.minLenght = CaptchaConstants.DEFAULT_LENGTH;
        this.maxLenght = CaptchaConstants.DEFAULT_LENGTH;
        this.type = type;
    }

    public TextProducerBuilder setLenght(int minLenght, int maxLenght) {
        this.minLenght = minLenght;
        this.maxLenght = maxLenght;
        return this;
    }

    public TextProducerBuilder setMinLenght(int minLenght) {
        this.minLenght = minLenght;
        return this;
    }

    public TextProducerBuilder setMaxLenght(int maxLenght) {
        this.maxLenght = maxLenght;
        return this;
    }

    @Override
    public TextProducer create() {
        switch (type) {
            case ALPHANUMERIC:
                return new AlphanumericTextProducer(minLenght, maxLenght);
            case REDUCED_ALPHANUMERIC:
                return new ReducedAlphanumericTextProducer(minLenght,
                    maxLenght);
            case CHINESE:
                return new ChineseTextProducer(minLenght, maxLenght);
            case ARABIC:
                return new ArabicTextProducer(minLenght, maxLenght);
            case NUMBERS:
                return new NumbersProducer(minLenght, maxLenght);
            case LETTERS:
                return new LetterTextProducer(minLenght, maxLenght);
            case LETTERS_SPECIAL:
                return new SpecialLetterTextProducer(minLenght, maxLenght);
            case NUMBERS_SPECIAL:
                return new SpecialNumbersProducer(minLenght, maxLenght);
            case ALPHANUMERIC_SPECIAL:
                return new SpecialAlphanumericTextProducer(minLenght,
                    maxLenght);
            default:
                throw new IllegalArgumentException("TextProducer not found: 
                    " + type.name());
        }
    }
}
```

Listing A.77: be.hogent.captchabuilder.elementcreator.renderer.gimpy.AbstractGimpyRenderer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.renderer.gimpy;

import be.hogent.captchabuilder.util.ColorRangeRGBA;

/**
 * AbstractGimpyRenderer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.6
 * @version 1.1.0
 */
public abstract class AbstractGimpyRenderer implements GimpyRenderer {

    protected double d1;
    protected double d2;
    protected ColorRangeRGBA colorRange1;
    protected ColorRangeRGBA colorRange2;

    protected AbstractGimpyRenderer(double d1, double d2, ColorRangeRGBA
        colorRange1, ColorRangeRGBA colorRange2) {
        this.d1 = d1;
        this.d2 = d2;
        this.colorRange1 = colorRange1;
        this.colorRange2 = colorRange2;
    }
}
```

Listing A.78: be.hogent.captchabuilder.elementcreator.renderer.gimpy.BlockGimpyRenderer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.renderer.gimpy;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import be.hogent.captchabuilder.util.ImageUtil;
import java.awt.image.BufferedImage;
import com.jhlabs.image.BlockFilter;

/**
 * BlockGimpyRenderer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.6
 * @version 1.1.0
 */
public class BlockGimpyRenderer extends AbstractGimpyRenderer {

    public BlockGimpyRenderer(double d1, double d2, ColorRangeRGBA
        colorRange1, ColorRangeRGBA colorRange2) {
        super(d1, d2, colorRange1, colorRange2);
    }

    @Override
    public void gimp(BufferedImage image) {
        BlockFilter filter = new BlockFilter();
        filter.setBlockSize((int) d1);
        ImageUtil.applyFilter(image, filter);
    }
}
```

Listing A.79: be.hogent.captchabuilder.elementcreator.renderer.gimpy.DropShadowGimpyRendere

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 *   copy
 * of this software and associated documentation files (the "Software"), to
 *   deal
 * in the Software without restriction, including without limitation the
 *   rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
 *   in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
 *   OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
 *   THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 *   FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.renderer.gimpy;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import be.hogent.captchabuilder.util.ImageUtil;
import java.awt.image.BufferedImage;
import com.jhlabs.image.ShadowFilter;

/**
 * DropShadowGimpyRenderer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.6
 * @version 1.1.0
 */
public class DropShadowGimpyRenderer extends AbstractGimpyRenderer {

    protected DropShadowGimpyRenderer(double d1, double d2, ColorRangeRGBA
        colorRange1, ColorRangeRGBA colorRange2) {
        super(d1, d2, colorRange1, colorRange2);
    }

    @Override
    public void gimp(BufferedImage image) {
        ShadowFilter sFilter = new ShadowFilter();
        sFilter.setRadius((int) d1);
        sFilter.setOpacity((int) d2);
        sFilter.setShadowColor(colorRange1.getRandomColorInRange().getRGB())
            ;
        ImageUtil.applyFilter(image, sFilter);
    }
}
```

Listing A.80: be.hogent.captchabuilder.elementcreator.renderer.gimpy.FishEyeGimpyRenderer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.renderer.gimpy;


import be.hogent.captchabuilder.util.ColorRangeRGBA;
import java.awt.image.BufferedImage;

import java.awt.Graphics2D;


/**
 * StretchGimpyRenderer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.6
 * @version 1.1.0
 */
public class FishEyeGimpyRenderer extends AbstractGimpyRenderer {

    public FishEyeGimpyRenderer(double d1, double d2, ColorRangeRGBA
        colorRange1, ColorRangeRGBA colorRange2) {
        super(d1, d2, colorRange1, colorRange2);
    }

    @Override
    public void gimp(BufferedImage image) {
        int height = image.getHeight();
        int width = image.getWidth();

        int hstripes = (int) (height / d1);
        int vstripes = (int) (width / d2);

        // Calculate space between lines
        int hspace = height / (hstripes + 1);
        int vspace = width / (vstripes + 1);

        Graphics2D graph = (Graphics2D) image.getGraphics();
        // Draw the horizontal stripes
        for (int i = hspace; i < height; i = i + hspace) {
            graph.setColor(colorRange1.getRandomColorInRange());
            graph.drawLine(0, i, width, i);
        }

        // Draw the vertical stripes
        for (int i = vspace; i < width; i = i + vspace) {
            graph.setColor(colorRange2.getRandomColorInRange());
            graph.drawLine(i, 0, i, height);
        }

        // Create a pixel array of the original image.
        // we need this later to do the operations on..
        int pix[] = new int[height * width];
        int j = 0;

        for (int j1 = 0; j1 < width; j1++) {
            for (int k1 = 0; k1 < height; k1++) {
                pix[j] = image.getRGB(j1, k1);
                j++;
            }
        }

        double distance = ranInt(width / 4, width / 3);

        // put the distortion in the (dead) middle
        int wMid = image.getWidth() / 2;
        int hMid = image.getHeight() / 2;

        // again iterate over all pixels..
        for (int x = 0; x < image.getWidth(); x++) {
            for (int y = 0; y < image.getHeight(); y++) {

                int relX = x - wMid;
                int relY = y - hMid;

                double d1 = Math.sqrt(relX * relX + relY * relY);
                if (d1 < distance) {

                    int j2 = wMid
                            + (int) (((fishEyeFormula(d1 / distance) *
                                distance) / d1) * (x - wMid));
                    int k2 = hMid
                            + (int) (((fishEyeFormula(d1 / distance) *
                                distance) / d1) * (y - hMid));
                    image.setRGB(x, y, pix[j2 * height + k2]);
                }
            }
        }

        graph.dispose();
    }

    private final int ranInt(int i, int j) {
        double d = Math.random();
        return (int) (i + ((j - i) + 1) * d);
    }

    private final double fishEyeFormula(double s) {
        // implementation of:
        // g(s) = - (3/4)s3 + (3/2)s2 + (1/4)s, with s from 0 to 1.
        if (s < 0.0D) {
            return 0.0D;
        }
        if (s > 1.0D) {
            return s;
        }

        return -0.75D * s * s * s + 1.5D * s * s + 0.25D * s;
    }
}
```

Listing A.81: be.hogent.captchabuilder.elementcreator.renderer.gimpy.GimpyRenderer

```
/*
```

```
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.renderer.gimpy;

import java.awt.image.BufferedImage;

/**
 * GimpyRenderer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.6
 * @version 1.0.7
 */
public interface GimpyRenderer {
    public void gimp(BufferedImage image);
}
```

Listing A.82: be.hogent.captchabuilder.elementcreator.renderer.gimpy.GimpyRendererBuilder

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.renderer.gimpy;

import be.hogent.captchabuilder.elementcreator.CaptchaElementCreatorBuilder;
import be.hogent.captchabuilder.util.ColorRangeRGBA;
import be.hogent.captchabuilder.util.enums.renderer.GimpyRendererType;

/**
 * GimpyRendererBuilder.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.6
 * @version 1.1.0
 */
public class GimpyRendererBuilder implements CaptchaElementCreatorBuilder {

    private double d1;
    private double d2;
    private ColorRangeRGBA colorRange1;
    private ColorRangeRGBA colorRange2;
    private GimpyRendererType type;

    public GimpyRendererBuilder(GimpyRendererType type) {
        this.colorRange1 = new ColorRangeRGBA(211, 211, 211);
        this.colorRange2 = new ColorRangeRGBA(169, 169, 169);

        this.d1 = 3.0;
        this.d2 = 75;
        this.type = type;
        if (type.equals(GimpyRendererType.STRETCH)) {
            this.d2 = 3.0;
        }
        if (type.equals(GimpyRendererType.RIPPLE)) {
            this.d1 = 2.6;
            this.d2 = 1.7;
        }
    }

    public GimpyRendererBuilder setD1(double d1) {
        this.d1 = d1;
        return this;
    }

    public GimpyRendererBuilder setD2(double d2) {
        this.d2 = d2;
        return this;
    }

    public GimpyRendererBuilder setColorRange1(ColorRangeRGBA colorRange1) {
        this.colorRange1 = colorRange1;
        return this;
    }

    public GimpyRendererBuilder setColorRange2(ColorRangeRGBA  colorRange2)
        {
        this.colorRange2 = colorRange2;
        return this;
    }

    @Override
    public GimpyRenderer create() {
        switch (type) {
            case BLOCK:
                return new BlockGimpyRenderer(d1, d2, colorRange1,
                    colorRange2);
            case DROPSHADOW:
                return new DropShadowGimpyRenderer(d1, d2, colorRange1,
                    colorRange2);
            case FISHEYE:
                return new FishEyeGimpyRenderer(d1, d2, colorRange1,
                    colorRange2);
            case RIPPLE:
                return new RippleGimpyRenderer(d1, d2, colorRange1,
                    colorRange2);
            case SHEAR:
                return new ShearGimpyRenderer(d1, d2, colorRange1,
                    colorRange2);
            case STRETCH:
                return new StretchGimpyRenderer(d1, d2, colorRange1,
                    colorRange2);
            default:
                throw new IllegalArgumentException("GimpyRenderer␣not␣found:
                    ␣" + type.name());
        }
    }
}
```

Listing A.83: be.hogent.captchabuilder.elementcreator.renderer.gimpy.RippleGimpyRenderer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.renderer.gimpy;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import be.hogent.captchabuilder.util.ImageUtil;
import com.jhlabs.image.RippleFilter;
import com.jhlabs.image.TransformFilter;
import java.awt.image.BufferedImage;

/**
 * RippleGimpyRenderer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.6
 * @version 1.1.0
 */
public class RippleGimpyRenderer extends AbstractGimpyRenderer {

    public RippleGimpyRenderer(double d1, double d2, ColorRangeRGBA
        colorRange1, ColorRangeRGBA colorRange2) {
        super(d1, d2, colorRange1, colorRange2);
    }

    @Override
    public void gimp(BufferedImage image) {
        RippleFilter filter = new RippleFilter();
        filter.setWaveType(RippleFilter.SINGLEFRAME);
        filter.setXAmplitude(d1);
        filter.setYAmplitude(d2);
        filter.setXWavelength((5.77)*d1);
        filter.setYWavelength((2.94)*d2);

        filter.setEdgeAction(TransformFilter.RANDOMPIXELORDER);
        ImageUtil.applyFilter(image, filter);
    }
}
```

Listing A.84: be.hogent.captchabuilder.elementcreator.renderer.gimpy.ShearGimpyRenderer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.renderer.gimpy;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import java.awt.image.BufferedImage;

import java.awt.Graphics2D;
import java.util.Random;

/**
 * ShearGimpyRenderer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.6
 * @version 1.1.0
 */
public class ShearGimpyRenderer extends AbstractGimpyRenderer {

    private Random random;
    public ShearGimpyRenderer(double d1, double d2, ColorRangeRGBA
        colorRange1, ColorRangeRGBA colorRange2) {
        super(d1, d2, colorRange1, colorRange2);
        this.random = CaptchaConstants.RANDOM;
    }

    @Override
    public void gimp(BufferedImage bi) {
        Graphics2D g = bi.createGraphics();
        shearX(g, bi.getWidth(), bi.getHeight());
        shearY(g, bi.getWidth(), bi.getHeight());
        g.dispose();
    }

    private void shearX(Graphics2D g, int w1, int h1) {

        int period = random.nextInt(10) + 5;

        boolean borderGap = true;
        int frames = 15;
        int phase = random.nextInt(5) + 2;

        for (int i = 0; i < h1; i++) {
            double d = (period >> 1)
                    * Math.sin((double) i / (double) period
                    + (6.2831853071795862D * phase) / frames);
            g.copyArea(0, i, w1, 1, (int) d, 0);
            if (borderGap) {
                g.setColor(colorRange1.getRandomColorInRange());
                g.drawLine((int) d, i, 0, i);
                g.drawLine((int) d + w1, i, w1, i);
            }
        }
    }

    private void shearY(Graphics2D g, int w1, int h1) {
        int period = random.nextInt(30) + 10; // 50;

        boolean borderGap = true;
        int frames = 15;
        int phase = 7;
        for (int i = 0; i < w1; i++) {
            double d = (period >> 1)
                    * Math.sin((float) i / period
                    + (6.2831853071795862D * phase) / frames);
            g.copyArea(i, 0, 1, h1, 0, (int) d);
            if (borderGap) {
                g.setColor(colorRange1.getRandomColorInRange());
                g.drawLine(i, (int) d, i, 0);
                g.drawLine(i, (int) d + h1, i, h1);
            }
        }
    }
}
```

Listing A.85: be.hogent.captchabuilder.elementcreator.renderer.gimpy.StretchGimpyRenderer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.renderer.gimpy;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import java.awt.image.BufferedImage;

import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;

/**
 * StretchGimpyRenderer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.6
 * @version 1.1.0
 */
public class StretchGimpyRenderer extends AbstractGimpyRenderer {

    public StretchGimpyRenderer(double d1, double d2, ColorRangeRGBA
        colorRange1, ColorRangeRGBA colorRange2) {
        super(d1, d2, colorRange1, colorRange2);
    }

    @Override
    public void gimp(BufferedImage image) {
        Graphics2D g = image.createGraphics();
        AffineTransform at = new AffineTransform();
        at.scale(d1, d2);
        g.drawRenderedImage(image, at);
    }
}
```

Listing A.86: be.hogent.captchabuilder.elementcreator.renderer.text.AbstractWordRenderer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.renderer.text;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.font.FontRenderContext;
import java.awt.image.BufferedImage;
import java.util.ArrayList;
import java.util.List;

/**
 * AbstractWordRenderer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.3
 * @version 1.1.0
 */
public abstract class AbstractWordRenderer implements WordRenderer {

    public static final ColorRangeRGBA DEFAULT_COLOR_RANGE;
    public static final List<Font> DEFAULT_FONTS = new ArrayList<>();

    static {
        DEFAULT_COLOR_RANGE = new ColorRangeRGBA(0);
        DEFAULT_FONTS.add(new Font("Arial", Font.BOLD, 40));
//        DEFAULT_FONTS.add(new Font("Courier", Font.BOLD, 40));
    }
    protected ColorRangeRGBA colorRange;
    protected List<Font> fonts;
    private double xOffset;
    private double yOffset;
    protected float strokeWidth;
    protected Graphics2D g;
    protected FontRenderContext frc;

    /**
     * Build a
     * <code>WordRenderer</code> using the given
     * <code>Color</code>s and
     * <code>Font</code>s.
     *
     * @param colorRange
     * @param fonts
     */
    public AbstractWordRenderer(ColorRangeRGBA colorRange, List<Font> fonts,
         double xOffset, double yOffset, float strokeWidth) {
        this.colorRange = colorRange;
        this.fonts = fonts;
        this.xOffset = xOffset;
        this.yOffset = yOffset;
        this.strokeWidth = strokeWidth;
    }

    /**
     * Render a word onto a BufferedImage.
     *
     * @param word The word to be rendered.
     * @param image The BufferedImage onto which the word will be painted.
     */
    protected void preRender(BufferedImage image) {
        g = image.createGraphics();

        RenderingHints hints = new RenderingHints(
                RenderingHints.KEY_ANTIALIASING,
                RenderingHints.VALUE_ANTIALIAS_ON);
        hints.add(new RenderingHints(RenderingHints.KEY_RENDERING,
                RenderingHints.VALUE_RENDER_QUALITY));
        g.setRenderingHints(hints);

        frc = g.getFontRenderContext();
    }

    protected int getXBaseline(BufferedImage image) {
        return (int) Math.round(image.getWidth() * xOffset);
    }

    protected int getYBaseline(BufferedImage image) {
        return image.getHeight() - (int) Math.round(image.getHeight() *
            yOffset);
    }
    protected Font getRandomFont() {
        return (Font) getRandomObject(fonts);
    }

    public Object getRandomObject(List<? extends Object> objs) {
        if (objs.size() == 1) {
            return objs.get(0);
        }

        int i = CaptchaConstants.RANDOM.nextInt(objs.size());
        return objs.get(i);
    }
}
```

Listing A.87: be.hogent.captchabuilder.elementcreator.renderer.text.ColoredEdgesWordRenderer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.renderer.text;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import java.awt.BasicStroke;
import java.awt.Font;
import java.awt.Shape;
import java.awt.font.TextAttribute;
import java.awt.font.TextLayout;
import java.awt.geom.AffineTransform;
import java.awt.image.BufferedImage;
import java.text.AttributedCharacterIterator;
import java.text.AttributedString;
import java.util.List;

/**
```

```
 * ColoredEdgesWordRenderer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.3
 * @version 1.1.0
 */
public class ColoredEdgesWordRenderer extends AbstractWordRenderer {

    public ColoredEdgesWordRenderer(ColorRangeRGBA colorRange, List<Font>
        fonts, double xOffset, double yOffset, float strokeWidth) {
        super(colorRange, fonts, xOffset, yOffset, strokeWidth);
    }

    @Override
    public void render(String word, BufferedImage bi) {
        preRender(bi);
        int xBaseline = getXBaseline(bi);
        int yBaseline = getYBaseline(bi);

        AttributedString as = new AttributedString(word);
        as.addAttribute(TextAttribute.FONT, getRandomFont());
        AttributedCharacterIterator aci = as.getIterator();

        TextLayout tl = new TextLayout(aci, frc);

        Shape shape = tl.getOutline(AffineTransform.getTranslateInstance(
            xBaseline, yBaseline));

        g.setColor(colorRange.getRandomColorInRange());
        g.setStroke(new BasicStroke(strokeWidth));

        g.draw(shape);
    }
}
```

Listing A.88: be.hogent.captchabuilder.elementcreator.renderer.text.DefaultWordRenderer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */

package be.hogent.captchabuilder.elementcreator.renderer.text;

import be.hogent.captchabuilder.util.ColorRangeRGBA;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import java.awt.Font;
import java.awt.font.GlyphVector;
import java.awt.image.BufferedImage;
import java.util.List;

/**
 * DefaultWordRenderer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.3
 * @version 1.1.0
 */
public class DefaultWordRenderer extends AbstractWordRenderer {

    public DefaultWordRenderer(ColorRangeRGBA colorRange, List<Font> fonts,
        double xOffset, double yOffset, float strokeWidth) {
        super(colorRange, fonts, xOffset, yOffset, strokeWidth);
    }

    @Override
    public void render(String word, BufferedImage bi) {
        preRender(bi);
        int xBaseline = getXBaseline(bi);
        int yBaseline =  getYBaseline(bi);

        char[] chars = new char[1];
        for (char c : word.toCharArray()) {
            chars[0] = c;

            g.setColor(colorRange.getRandomColorInRange());

            int choiceFont = CaptchaConstants.RANDOM.nextInt(fonts.size());
            Font font = fonts.get(choiceFont);
            g.setFont(font);

            GlyphVector gv = font.createGlyphVector(frc, chars);
            g.drawChars(chars, 0, chars.length, xBaseline, yBaseline);

            int width = (int) gv.getVisualBounds().getWidth();
```

```
                xBaseline = xBaseline + width;
            }
        }


}
```

Listing A.89: be.hogent.captchabuilder.elementcreator.renderer.text.WordRenderer

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.elementcreator.renderer.text;

import java.awt.image.BufferedImage;

/**
 * WordRenderer.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.4
 * @version 1.0.7
 */
public interface WordRenderer {

    public void render(String word, BufferedImage image);
}
```

Listing A.90: be.hogent.captchabuilder.elementcreator.renderer.text.WordRendererBuilder

```
/*
 * The MIT License
 *
 * Copyright 2013 piva.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */

package be.hogent.captchabuilder.elementcreator.renderer.text;

import be.hogent.captchabuilder.elementcreator.CaptchaElementCreatorBuilder;
import be.hogent.captchabuilder.util.ColorRangeRGBA;
import be.hogent.captchabuilder.util.enums.CaptchaConstants;
import be.hogent.captchabuilder.util.enums.renderer.WordRendererType;
import java.awt.Font;
import java.util.List;

/**
 * WordRendererBuilder.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.3
 * @version 1.1.0
 */
public class WordRendererBuilder implements CaptchaElementCreatorBuilder {
    private ColorRangeRGBA colorRange;
    private List<Font> fonts;
    private double xOffset;
    private double yOffset;
    private float strokeWidth;
    private WordRendererType type;
    public WordRendererBuilder(WordRendererType type) {
        this.strokeWidth = CaptchaConstants.DEFAULT_STROKE_WIDTH;
        this.yOffset = CaptchaConstants.DEFAULT_YOFFSET;
        this.xOffset = CaptchaConstants.DEFAULT_XOFFSET;
        this.fonts = AbstractWordRenderer.DEFAULT_FONTS;
        this.colorRange = AbstractWordRenderer.DEFAULT_COLOR_RANGE;
        this.type = type;
    }

    public WordRendererBuilder setColorRange(ColorRangeRGBA colorRange) {
        this.colorRange = colorRange;
        return this;
    }

    public WordRendererBuilder setFonts(List<Font> fonts) {
        this.fonts = fonts;
        return this;
    }

    public WordRendererBuilder setXOffset(double xOffset) {
        this.xOffset = xOffset;
        return this;
    }

    public WordRendererBuilder setYOffset(double yOffset) {
        this.yOffset = yOffset;
        return this;
    }

    public WordRendererBuilder setStrokeWidth(float strokeWidth) {
        this.strokeWidth = strokeWidth;
        return this;
    }

    @Override
    public WordRenderer create() {
        switch (type) {
            case DEFAULT:
                return new DefaultWordRenderer(colorRange, fonts, xOffset,
                    yOffset, strokeWidth);
            case COLOREDEDGES:
                return new ColoredEdgesWordRenderer(colorRange, fonts,
                    xOffset, yOffset, strokeWidth);
            default:
                throw new IllegalArgumentException("WordRenderer not found: 
                    " + type.name());
        }
    }

}
```

Listing A.91: be.hogent.captchabuilder.util.enums.producer.BackgroundProducerType

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.util.enums.producer;

import be.hogent.captchabuilder.elementcreator.producer.background.
    BackgroundProducerBuilder;
import be.hogent.captchabuilder.elementcreator.producer.background.
    BackgroundProducer;

/**
 * BackgroundProducerType.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.4
 * @version 1.0.13
 */
public enum BackgroundProducerType {
    FLATCOLOR("Creates␣a␣background␣in␣a␣single␣color"),
    SQUIGGLES("Creates␣a␣squiggly␣background"),
    TRANSPARENT("Creates␣a␣transparent␣background"),
    TWOCOLORGRADIENT("Creates␣a␣two␣color␣horizontal␣gradient␣background");
    private String description;

    private BackgroundProducerType(String description) {
        this.description = description;
    }

    public String getDescription() {
        return description;
    }

    public BackgroundProducer getBackgroundProducer() {
        return new BackgroundProducerBuilder(this).create();
    }
}
```

Listing A.92: be.hogent.captchabuilder.util.enums.producer.BorderProducerType

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.util.enums.producer;

import be.hogent.captchabuilder.elementcreator.producer.border.
    BorderProducer;
import be.hogent.captchabuilder.elementcreator.producer.border.
    BorderProducerBuilder;

/**
 * BorderProducerType.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/18
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.12
 * @version 1.0.13
 */
public enum BorderProducerType {
    SOLID("Creates␣a␣solid␣border");
    private String description;

    private BorderProducerType(String description) {
        this.description = description;
    }
    public String getDescription() {
        return description;
    }

    public BorderProducer getBorderProducer() {
        return new BorderProducerBuilder(this).create();
    }
}
```

Listing A.93: be.hogent.captchabuilder.util.enums.producer.NoiseProducerType

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */

package be.hogent.captchabuilder.util.enums.producer;

import be.hogent.captchabuilder.elementcreator.producer.noise.
    NoiseProducerBuilder;
import be.hogent.captchabuilder.elementcreator.producer.noise.NoiseProducer;

/**
 * NoiseProducerType.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.5
 * @version 1.0.13
 */
public enum NoiseProducerType {
    CURVEDLINE("creates␣a␣curved␣line␣on␣the␣image␣to␣serve␣as␣noise"),
    STRAIGHTLINE("creates␣a␣straight␣line␣on␣the␣image␣to␣serve␣as␣noise");
    private String description;

    private NoiseProducerType(String description) {
        this.description = description;
    }

    public String getDescription() {
        return description;
    }

    public NoiseProducer getNoiseProducer() {
        return new NoiseProducerBuilder(this).create();
    }
}
```

Listing A.94: be.hogent.captchabuilder.util.enums.producer.TextProducerType

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.util.enums.producer;

import be.hogent.captchabuilder.elementcreator.producer.text.
    TextProducerBuilder;
import be.hogent.captchabuilder.elementcreator.producer.text.TextProducer;

/**
 * TextProducerType.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/14
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.1
 * @version 1.0.13
 */
public enum TextProducerType {

    ALPHANUMERIC("Generates alphanumeric strings"),
    REDUCED_ALPHANUMERIC("Generates reduced alphanumeric characterset 
        strings to prevent ambiguities"),
    CHINESE("Generates Chinese character strings"),
    ARABIC("Generates Chinese character strings"),
    NUMBERS("Generates number strings"),
    LETTERS("Generates normal character strings"),
    LETTERS_SPECIAL("Generates normal character combined with special 
        character strings"),
    NUMBERS_SPECIAL("Generates number strings combined with special 
        character strings"),
    ALPHANUMERIC_SPECIAL("Generates alphanumeric strings combined with 
        special character strings");
    private String desciption;

    private TextProducerType(String desciption) {
        this.desciption = desciption;
    }

    public TextProducer getTextProducer() {
        return new TextProducerBuilder(this).create();
    }

    public String getDescription() {
        return desciption;
    }

    @Override
    public String toString() {
        return name() + ": " + desciption;
    }
}
```

Listing A.95: be.hogent.captchabuilder.util.enums.renderer.GimpyRendererType

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.util.enums.renderer;

import be.hogent.captchabuilder.elementcreator.renderer.gimpy.
    GimpyRendererBuilder;
import be.hogent.captchabuilder.elementcreator.renderer.gimpy.GimpyRenderer;

/**
 * GimpyRendererType.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.3
 * @version 1.0.13
 */
public enum GimpyRendererType {
    BLOCK("Description:␣block"),
    DROPSHADOW("Description:␣dropshadow"),
    FISHEYE("Description:␣fish␣eye"),
    RIPPLE("Description:␣ripple"),
    SHEAR("Description:␣shear"),
    STRETCH("Description:␣stretch");
    private String description;

    private GimpyRendererType(String description) {
        this.description = description;
    }

    public String getDescription() {
        return description;
    }

    public GimpyRenderer getGimpyRenderer() {
        return new GimpyRendererBuilder(this).create();
    }
}
```

Listing A.96: be.hogent.captchabuilder.util.enums.renderer.WordRendererType

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
     copy
 * of this software and associated documentation files (the "Software"), to
     deal
 * in the Software without restriction, including without limitation the
     rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
     in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
     OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
     THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchabuilder.util.enums.renderer;

import be.hogent.captchabuilder.elementcreator.renderer.text.
    WordRendererBuilder;
import be.hogent.captchabuilder.elementcreator.renderer.text.WordRenderer;

/**
 * WordRendererType.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/04/16
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.3
 * @version 1.0.13
 */
public enum WordRendererType {

    COLOREDEDGES("Description"),
    DEFAULT("The␣default␣word␣renderer");
    private String desciption;

    private WordRendererType(String desciption) {
        this.desciption = desciption;
    }

    public WordRenderer getWordRenderer() {
        return new WordRendererBuilder(this).create();
    }

    public String getDescription() {
        return desciption;
    }

    @Override
    public String toString() {
        return name() + ": " + desciption;
    }
}
```

Listing A.97: be.hogent.captchasolvingnetwork.network.encog.util.PropagationType

```
/*
 * The MIT License
 *
 * Copyright 2013 Pieter Van Eeckhout.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
    copy
 * of this software and associated documentation files (the "Software"), to
    deal
 * in the Software without restriction, including without limitation the
    rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
    in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package be.hogent.captchasolvingnetwork.network.encog.util;

/**
 * PropagationType.java (UTF-8)
 *
 * usage and functionality here
 *
 * 2013/05/19
 *
 * @author Pieter Van Eeckhout <vaneeckhout.pieter@gmail.com>
 * @author Pieter Van Eeckhout <pieter.vaneeckhout.q1295@student.hogent.be>
 * @author Hogent StudentID <2000901295>
 * @since 1.0.0
 * @version 1.0.0
 */
public enum PropagationType {

    Backpropagation,
    ManhattanPropagation,
    ResilientPropagation,
    ScaledConjugateGradient;
}
```

Bibliography

Stephen Cobb. The Economics of Spam, 2003. URL http://spamhelp.whybot.com/articles/economics_of_spam.pdf.

Dennis W K Khong. An Economic Analysis of Spam Law. *Erasmus Law and Economics Review*, 1(February):23–45, 2004. URL http://www.eler.org/viewarticle.php?id=2.

Yasuharu Ukai and Toshihiko Takemura. Spam mails impede economic growth. *The Review of Socionetwork Strategies*, 1(1):14–22, March 2007. ISSN 1867-3236. doi: 10.1007/BF02981628. URL http://link.springer.com/10.1007/BF02981628.

Roger Stephen Young. *How Computers Work Processor and Main Memory*. 2001. URL http://www.fastchip.net/howcomputerswork/bookbpdf.pdf.

List of Figures

List of Tables

Listings