| | |
|---|---|
| **NAME** | JACOB JOHN |
| **REGISTER NO.** | 16BCE2205 |
| **E-MAIL** | jacob.john2016@vitstudent.ac.in |
| **COURSE** | Java Programming |


# *Digital Assignment 1*

## Sending Email in Java through Gmail Server

### Get the session object

The ***javax.mail.Session class*** provides two methods to get the object of session, *Session.getDefaultInstance()* method and *Session.getInstance()* method.

| Method | Description |
|---|---|
| ***public static Session getDefaultInstance(Properties props)*** | returns the default session. |
| ***public static Session getDefaultInstance(Properties props,Authenticator auth)*** | returns the default session. |
| ***public static Session getInstance(Properties props)*** | returns the new session. |
| ***public static Session getInstance(Properties props,Authenticator auth)*** | returns the new session. |

### Compose the message

The **javax.mail.Message class** provides methods to compose the message. But it is an abstract class so its subclass javax.mail.internet.MimeMessage class is mostly used.

To create the message, you need to pass session object in MimeMessage class constructor. For example:

```
MimeMessage message = new MimeMessage(session);
```

| Method | Description |
|---|---|
| ***public void setFrom(Address address)*** | is used to set the from header field. |
| ***public void addRecipient(Message.RecipientType type, Address address)*** | is used to add the given address to the recipient type. |
| ***public void addRecipients(Message.RecipientType type, Address[] addresses)*** | is used to add the given addresses to the recipient type. |
| ***public void setSubject(String subject)*** | is used to set the subject header field. |
| ***public void setText(String textmessage)*** | is used to set the text as the message content using text/plain MIME type. |
| ***public void setContent(Object msg, String contentType)*** | is used to set the content as the message content using given MIME type. |

For sending the email using JavaMail API, you need to load the two jar files:

- ***mail.jar***
- ***activation.jar***

download these jar files (or) go to the Oracle site to download the latest version.

## Resolving Authentication failure

Click on this link and click on turn on radio button to allow users to send mail from unknown location. https://www.google.com/settings/security/lesssecureapps

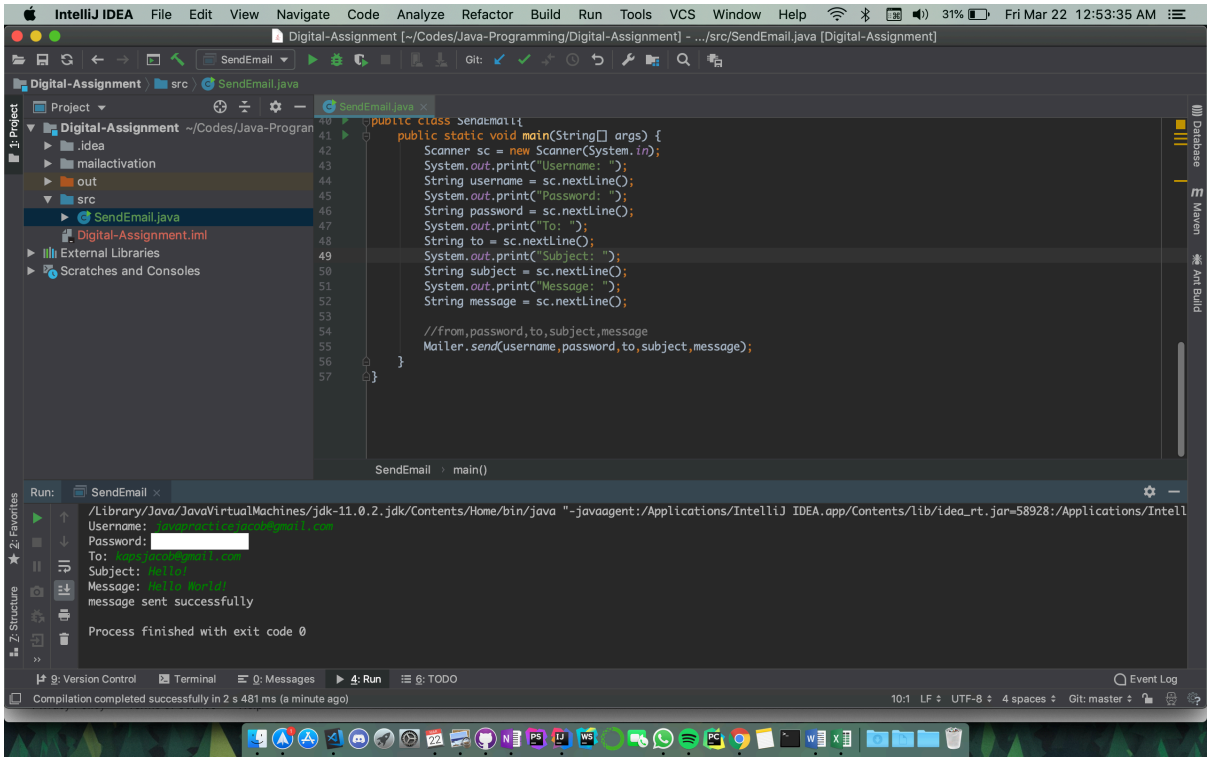
## Code

```
/*
 * Sending Email in Java through Gmail Server
 * Jar files: https://www.javatpoint.com/src/mail/mailactivation.zip
 * Source: https://www.javatpoint.com/example-of-sending-email-using-java-mail-
api-through-gmail-server
 */

import java.util.Properties;
import java.util.Scanner;
import javax.mail.*;
import javax.mail.internet.*;

class Mailer{
    public static void send(String from,String password,String to,String
sub,String msg){
        //Get properties object
        Properties props = new Properties();
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.socketFactory.port", "465");
        props.put("mail.smtp.socketFactory.class",
                "javax.net.ssl.SSLSocketFactory");
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.port", "465");
        //get Session
        Session session = Session.getDefaultInstance(props,
                new javax.mail.Authenticator() {
                    protected PasswordAuthentication getPasswordAuthentication()
{
                        return new PasswordAuthentication(from,password);
                    }
                });
        //compose message
        try {
            MimeMessage message = new MimeMessage(session);
            message.addRecipient(Message.RecipientType.TO,new
InternetAddress(to));
            message.setSubject(sub);
            message.setText(msg);
            //send message
            Transport.send(message);
            System.out.println("message sent successfully");
        } catch (MessagingException e) {throw new RuntimeException(e);}

    }
```

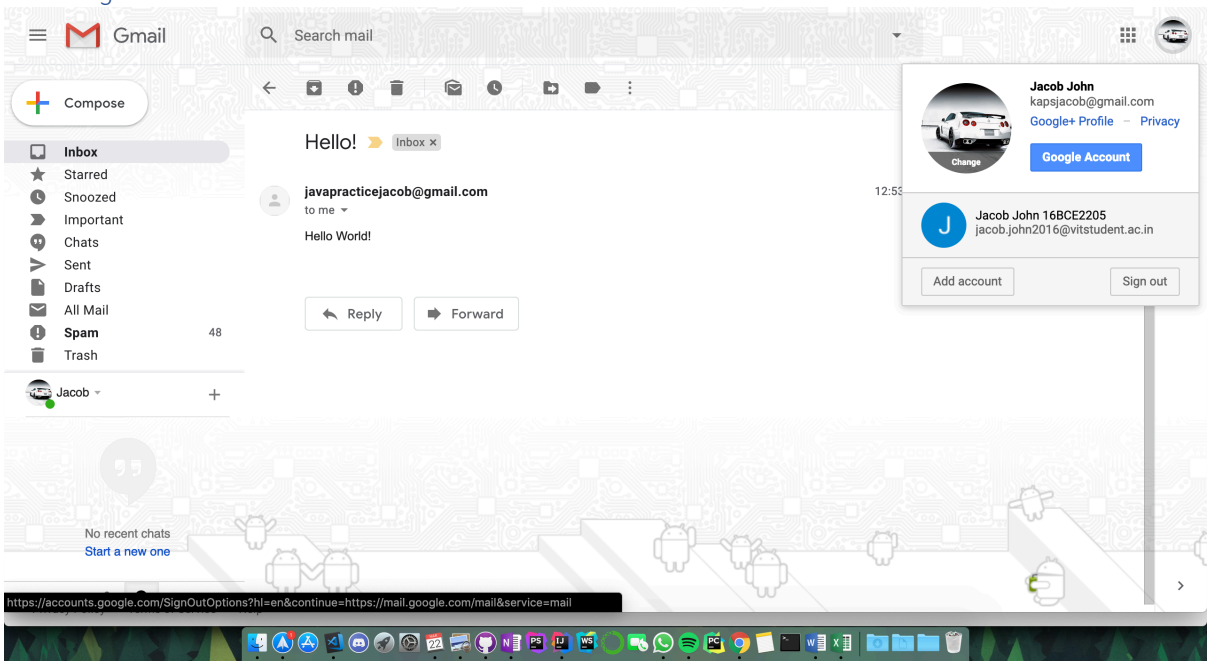
```
}

public class SendEmail{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Username: ");
        String username = sc.nextLine();
        System.out.print("Password: ");
        String password = sc.nextLine();
        System.out.print("To: ");
        String to = sc.nextLine();
        System.out.print("Subject: ");
        String subject = sc.nextLine();
        System.out.print("Message: ");
        String message = sc.nextLine();

        //from,password,to,subject,message
        Mailer.send(username,password,to,subject,message);
    }
}
```

## Output

*Sending the mail*



*Receiving mail*




## Language Detection using Apache OpenNLP

This is a Java project for Language Detection using Apache OpenNLP.

The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text. It supports the most common NLP tasks, such as language detection, tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing and coreference resolution.

### Language Detector

The Apache OpenNLP team announced the release of Language Detector Model 1.8.3 for Apache OpenNLP 1.8.3. The Language Detector Model can detect 103 languages and outputs ISO 639-3 codes.

This model is trained for and works well with longer texts that have at least 2 sentences or more from the same language.

According to https://www.apache.org/dist/opennlp/models/langdetect/1.8.3/README.txt

```
# Language Detector model for Apache OpenNLP #

The model were trained with the Leipzig corpus, which can be found here:
http://wortschatz.uni-leipzig.de/en/download/
The model can detect 103 languages in ISO 639-3 standard. The languages
are:

```
afr     Afrikaans
ara     Arabic
ast     Asturian
aze     Azerbaijani
bak     Bashkir
.
.
.
```
```

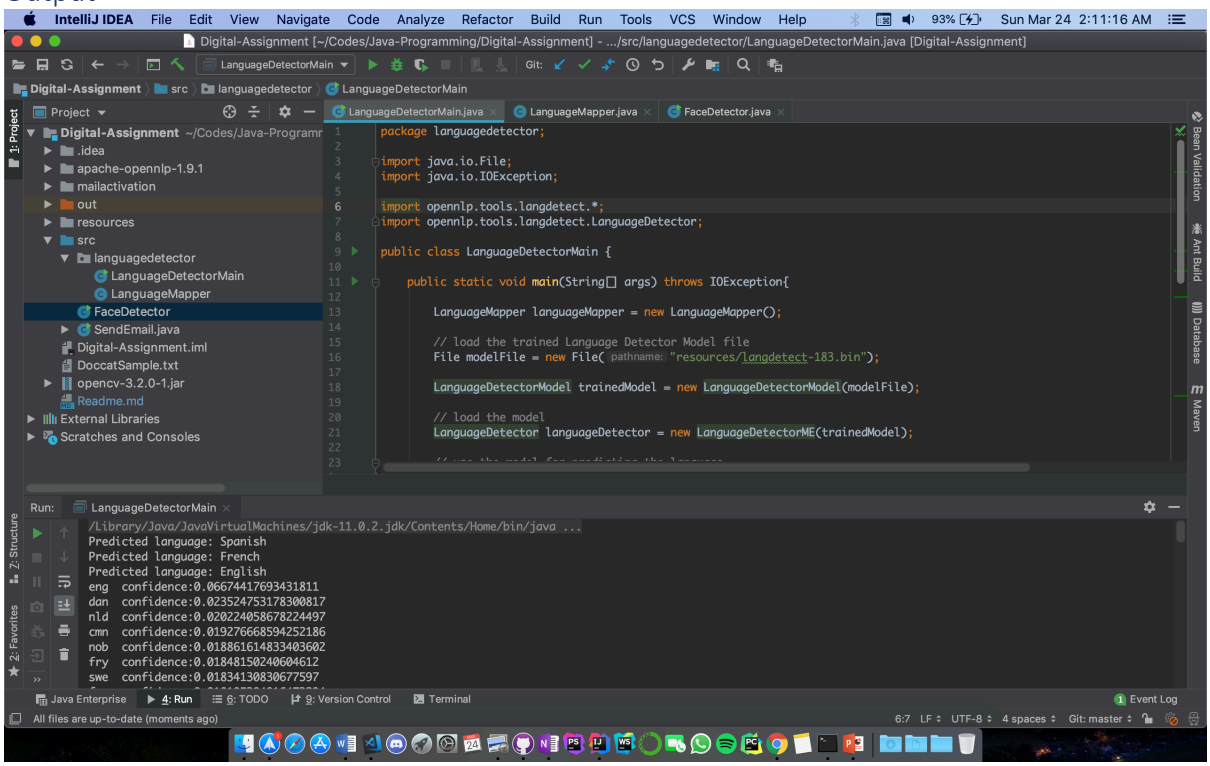
LanguageMapperMain.java

```
package languagedetector;

import java.io.File;
import java.io.IOException;

import opennlp.tools.langdetect.*;
import opennlp.tools.langdetect.LanguageDetector;

public class LanguageDetectorMain {

    public static void main(String[] args) throws IOException{

        LanguageMapper languageMapper = new LanguageMapper();

        // load the trained Language Detector Model file
        File modelFile = new File("resources/langdetect-183.bin");

        LanguageDetectorModel trainedModel = new
LanguageDetectorModel(modelFile);

        // load the model
        LanguageDetector languageDetector = new LanguageDetectorME(trainedModel);

        // use the model for predicting the language
        //Spanish
        Language[] languages = languageDetector.predictLanguages("Puedo darte
ejemplos de los métodos");

        System.out.println("Predicted language: "+
languageMapper.getLanguage(languages[0].getLang()));

        // French
        languages = languageDetector.predictLanguages("Je peux vous donner
quelques exemples de méthodes qui ont fonctionné pour moi.");

        System.out.println("Predicted language: "+
languageMapper.getLanguage(languages[0].getLang()));

        // English
        languages = languageDetector.predictLanguages("I can give you some
examples of methods that have worked for me.");

        System.out.println("Predicted language: "+
languageMapper.getLanguage(languages[0].getLang()));

        // confidence for rest of the languages
          for(Language language:languages){
               System.out.println(language.getLang()+"
confidence:"+language.getConfidence());
           }
    }
}
```

LanguageMapper.java

```java
package languagedetector;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;

public class LanguageMapper {

    public String getLanguage(String langCode) throws IOException {

        HashMap<String, String> langMap = readToHashmap();

        return langMap.get(langCode);

    }

    public HashMap<String, String> readToHashmap() throws IOException {
        HashMap<String, String> map = new HashMap<String, String>();
        BufferedReader in = new BufferedReader(new
FileReader("resources/language_mapping.txt"));
        String line = "";

        while ((line = in.readLine()) != null) {
            String parts[] = line.split("\t");
            map.put(parts[0], parts[1]);
        }
        in.close();

        return map;

    }
}
```

## Output




## Commons Math: The Apache Commons Mathematics Library

Commons Math is a library of lightweight, self-contained mathematics and statistics components addressing the most common problems not available in the Java programming language or Commons Lang.

## Statistics

The package org.apache.commons.math3.stat provides several tools for statistical computations. For example, to compute mean, standard deviation, and many more, we can use DescriptiveStatistics. In this package, we can find tools for computing the covariance, correlation, or to perform statistical tests (using TestUtils).

## Probabilities and Distributions

In core Java, Math.random() can be used for generating random values, but these values are uniformly distributed between 0 and 1.

Sometimes, we want to produce a random value using a more complex distribution. For this, we can use the framework provided by org.apache.commons.math3.distribution.

Here is how to generate random values according to the normal distribution with the mean of 10 and the standard deviation of 3:

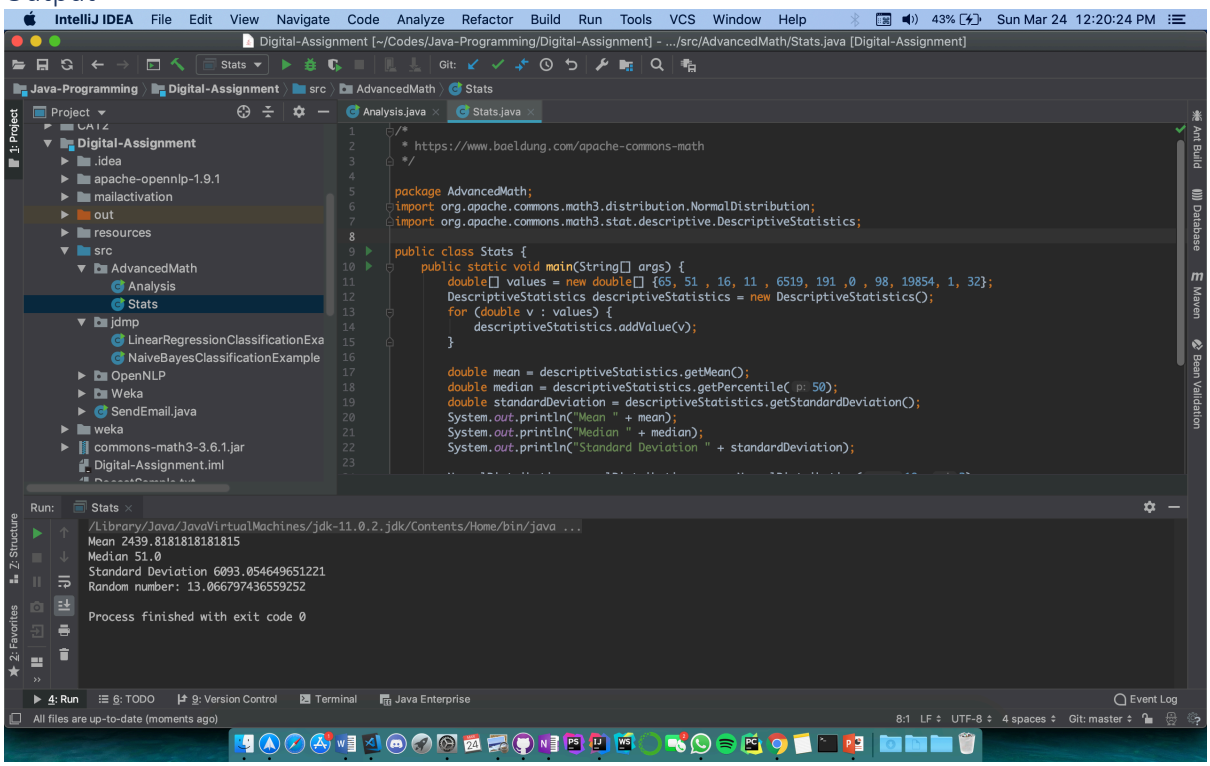
```
NormalDistribution normalDistribution = new NormalDistribution(10, 3);
double randomValue = normalDistribution.sample();
```

Or we can obtain the probability $P(X = x)$of getting a value for discrete distributions, or the cumulative probability $P(X <= x)$ for continuous distributions.

Code

```java
/*
 * https://www.baeldung.com/apache-commons-math
 */

package AdvancedMath;
import org.apache.commons.math3.distribution.NormalDistribution;
import org.apache.commons.math3.stat.descriptive.DescriptiveStatistics;

public class Stats {
    public static void main(String[] args) {
        double[] values = new double[] {65, 51 , 16, 11 , 6519, 191 ,0 , 98,
19854, 1, 32};
        DescriptiveStatistics descriptiveStatistics = new
DescriptiveStatistics();
        for (double v : values) {
            descriptiveStatistics.addValue(v);
        }

        double mean = descriptiveStatistics.getMean();
        double median = descriptiveStatistics.getPercentile(50);
        double standardDeviation = descriptiveStatistics.getStandardDeviation();
        System.out.println("Mean " + mean);
        System.out.println("Median " + median);
        System.out.println("Standard Deviation " + standardDeviation);

        NormalDistribution normalDistribution = new NormalDistribution(10, 3);
        double randomValue = normalDistribution.sample();
        System.out.println("Random number: " + randomValue);
    }
}
```

## Output




## Analysis

Analysis related functions and algorithms can be found in org.apache.commons.math3.analysis.

## Root Finding

A root is a value where a function has the value of 0. Commons-Math includes implementation of several root-finding algorithms.

Here, we try to find the root of v -> (v * v) – 2 :

```
UnivariateFunction function = v -> Math.pow(v, 2) - 2;
UnivariateSolver solver = new BracketingNthOrderBrentSolver(1.0e-12, 1.0e-8, 5);
double c = solver.solve(100, function, -10.0, 10.0, 0);
```

First, we start by defining the function, then we define the solver, and we set the desired accuracy. Finally, we call the solve() API.

The root-finding operation will be performed using several iterations, so it's a matter of finding a compromise between execution time and accuracy.

## Calculating Integrals

The integration works almost like root finding:

```
UnivariateFunction function = v -> v;
UnivariateIntegrator integrator = new SimpsonIntegrator(1.0e-12, 1.0e-8, 1, 32);
double i = integrator.integrate(100, function, 0, 10);
```

We start by defining a function, we choose an integrator among the available integration solutions existing, we set the desired accuracy, and finally, we integrate.

## Linear Algebra

If we have a linear system of equations under the form AX = B where A is a matrix of real numbers, and B a vector of real numbers – Commons Math provides structures to represent both the matrix and the vector, and also provide solvers to find the value of X:

```
RealMatrix a = new Array2DRowRealMatrix(
  new double[][] { { 2, 3, -2 }, { -1, 7, 6 }, { 4, -3, -5 } },
  false);
RealVector b = new ArrayRealVector(n
  ew double[] { 1, -2, 1 },
  false);

DecompositionSolver solver = new LUDecomposition(a).getSolver();

RealVector solution = solver.solve(b);
```

The case is pretty straightforward: we define a matrix a from an array of array of doubles, and a vector b from an array of a vector.

Then, we create an LUDecomposition which provides a solver for equations under the form AX = B. As its name states it, LUDecomposition relies on the LU decomposition, and thus works only with square matrices.

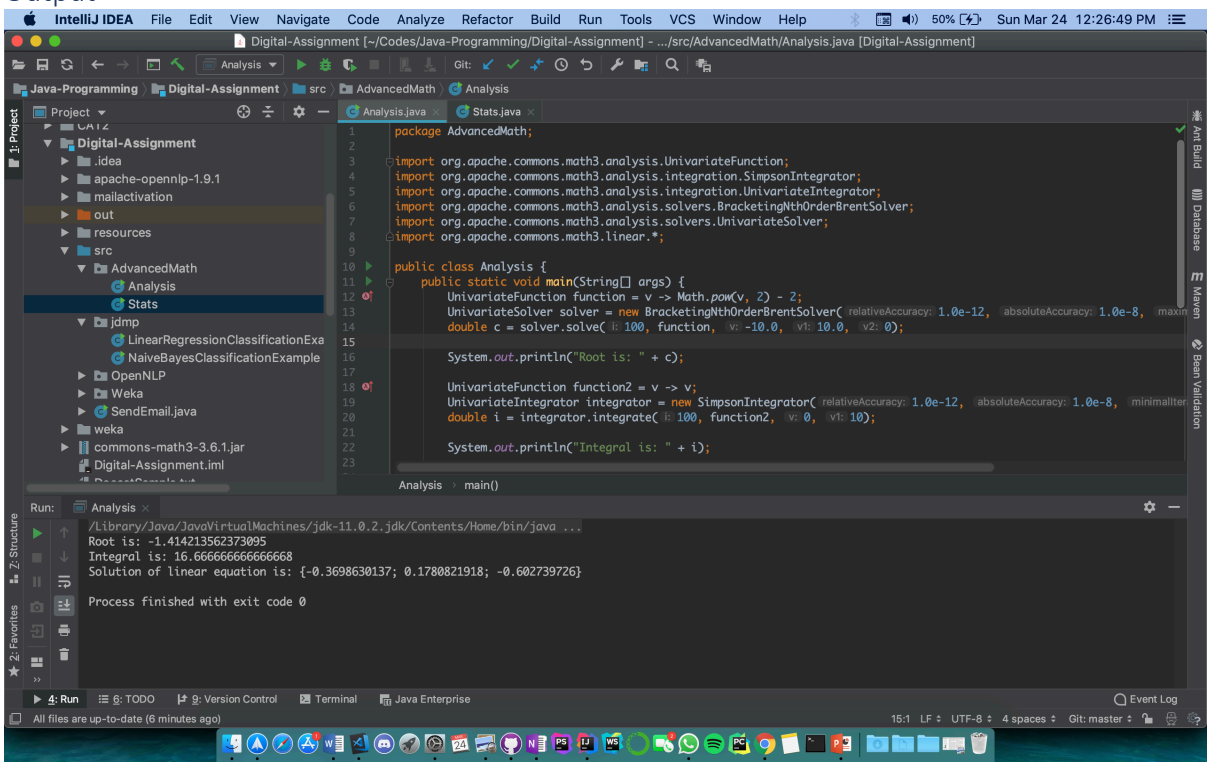
For other matrices, different solvers exist, usually solving the equation using the least square method.

## Code

```java
package AdvancedMath;

import org.apache.commons.math3.analysis.UnivariateFunction;
import org.apache.commons.math3.analysis.integration.SimpsonIntegrator;
import org.apache.commons.math3.analysis.integration.UnivariateIntegrator;
import org.apache.commons.math3.analysis.solvers.BracketingNthOrderBrentSolver;
import org.apache.commons.math3.analysis.solvers.UnivariateSolver;
import org.apache.commons.math3.linear.*;

public class Analysis {
    public static void main(String[] args) {
        UnivariateFunction function = v -> Math.pow(v, 2) - 2;
        UnivariateSolver solver = new BracketingNthOrderBrentSolver(1.0e-12,
1.0e-8, 5);
        double c = solver.solve(100, function, -10.0, 10.0, 0);

        System.out.println("Root is: " + c);

        UnivariateFunction function2 = v -> v;
        UnivariateIntegrator integrator = new SimpsonIntegrator(1.0e-12, 1.0e-8,
1, 32);
        double i = integrator.integrate(100, function2, 0, 10);

        System.out.println("Integral is: " + i);

        RealMatrix a = new Array2DRowRealMatrix(
                new double[][] { { 2, 3, -2 }, { -1, 7, 6 }, { 4, -3, -5 } },
                false);
        RealVector b = new ArrayRealVector(new double[] { 1, -2, 1 }, false);
```

```
        DecompositionSolver solver2 = new LUDecomposition(a).getSolver();

        RealVector solution = solver2.solve(b);

        System.out.println("Solution of linear equation is: " + solution);
    }
}
```

## Output




## Sentence Tokenization

To tokenize a sentence using the SimpleTokenizer class, you need to –

1. Create an object of the respective class.
2. Tokenize the sentence using the tokenize() method.
3. Print the tokens.

Following are the steps to be followed to write a program which tokenizes the given raw text.

**Step 1** – Instantiating the respective class

In both the classes, there are no constructors available to instantiate them. Therefore, we need to create objects of these classes using the static variable INSTANCE.

```
//Instantiating SimpleTokenizer class
SimpleTokenizer simpleTokenizer = SimpleTokenizer.INSTANCE;
```

**Step 2** – Tokenize the sentences

Both these classes contain a method called tokenize(). This method accepts a raw text in String format. On invoking, it tokenizes the given String and returns an array of Strings (tokens).

```
//Tokenizing the given sentence
String tokens[] = simpleTokenizer.tokenize(sentence);
```

**Step 3** – Print the tokens

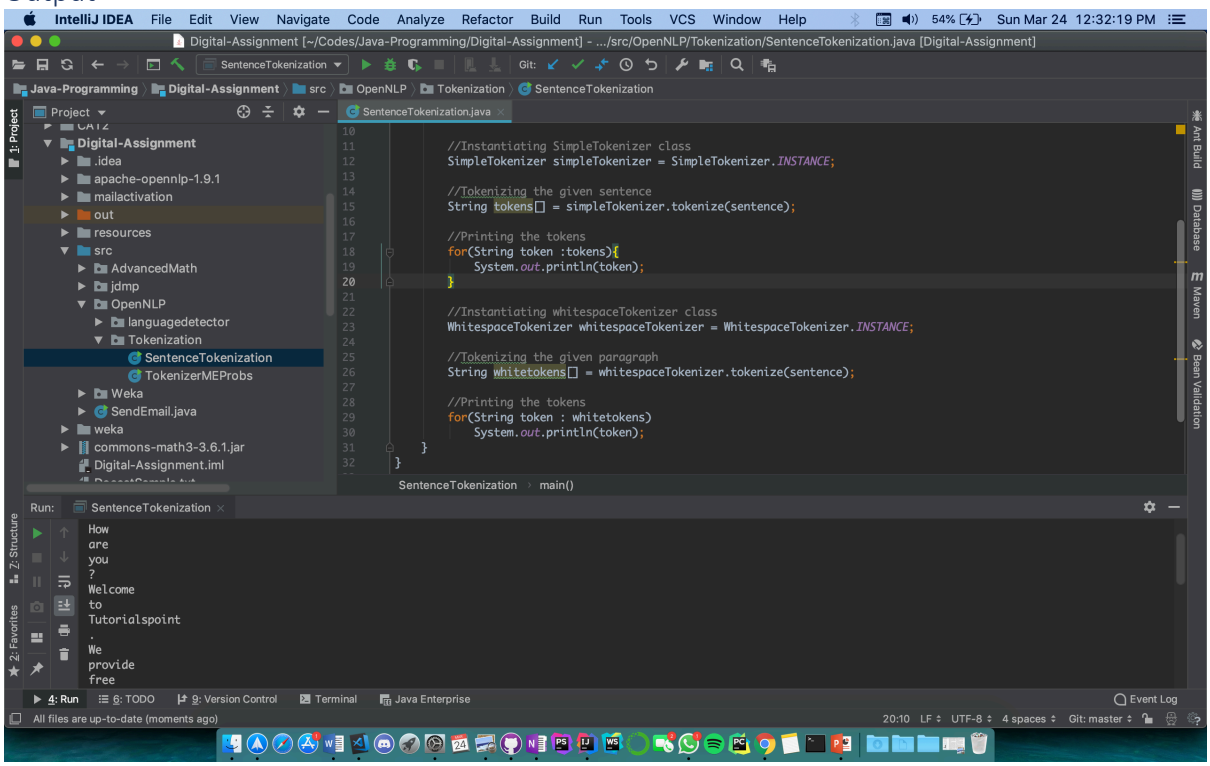
After tokenizing the sentence, you can print the tokens using for loop, as shown below.

```
//Printing the tokens
for(String token :tokens){
    System.out.println(token);
}
```

## Code

```
package OpenNLP.Tokenization;

import opennlp.tools.tokenize.SimpleTokenizer;
import opennlp.tools.tokenize.WhitespaceTokenizer;

public class SentenceTokenization {
    public static void main(String[] args) {
        String sentence = "Hi. How are you? Welcome to Tutorialspoint. "
                + "We provide free tutorials on various technologies";

        //Instantiating SimpleTokenizer class
        SimpleTokenizer simpleTokenizer = SimpleTokenizer.INSTANCE;

        //Tokenizing the given sentence
        String tokens[] = simpleTokenizer.tokenize(sentence);

        //Printing the tokens
        for(String token :tokens){
            System.out.println(token);
        }

        //Instantiating whitespaceTokenizer class
        WhitespaceTokenizer whitespaceTokenizer = WhitespaceTokenizer.INSTANCE;

        //Tokenizing the given paragraph
        String whitetokens[] = whitespaceTokenizer.tokenize(sentence);

        //Printing the tokens
        for(String token : whitetokens)
            System.out.println(token);
    }
}
```

## Output




## Tokenization with a predefined model

OpenNLP also uses a predefined model, a file named de-token.bin, to tokenize the sentences. It is trained to tokenize the sentences in a given raw text.

The TokenizerME class of the opennlp.tools.tokenizer package is used to load this model, and tokenize the given raw text using OpenNLP library. To do so, you need to −

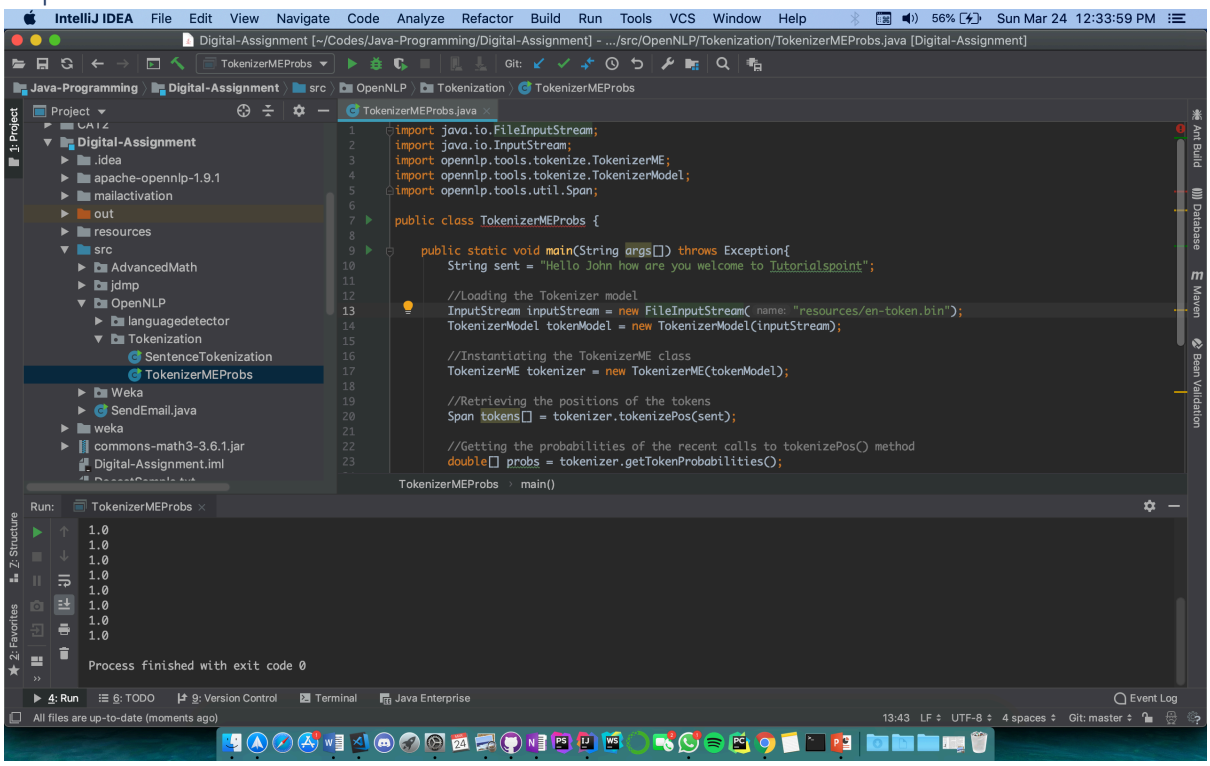
- Load the en-token.bin model using the TokenizerModel class.
- Instantiate the TokenizerME class.
- Tokenize the sentences using the tokenize() method of this class.

## Code

```
import java.io.FileInputStream;
import java.io.InputStream;
import opennlp.tools.tokenize.TokenizerME;
import opennlp.tools.tokenize.TokenizerModel;
import opennlp.tools.util.Span;

public class TokenizerMEProbs {

    public static void main(String args[]) throws Exception{
        String sent = "Hello John how are you welcome to Tutorialspoint";

        //Loading the Tokenizer model
        InputStream inputStream = new FileInputStream("resources/en-token.bin");
        TokenizerModel tokenModel = new TokenizerModel(inputStream);

        //Instantiating the TokenizerME class
        TokenizerME tokenizer = new TokenizerME(tokenModel);

        //Retrieving the positions of the tokens
```

```
        Span tokens[] = tokenizer.tokenizePos(sent);

        //Getting the probabilities of the recent calls to tokenizePos() method
        double[] probs = tokenizer.getTokenProbabilities();

        //Printing the spans of tokens
        for(Span token : tokens)
            System.out.println(token +" "+sent.substring(token.getStart(),
token.getEnd()));
        System.out.println("  ");
        for(int i = 0; i<probs.length; i++)
            System.out.println(probs[i]);
    }
}
```

## Ouput




## Clustering in java

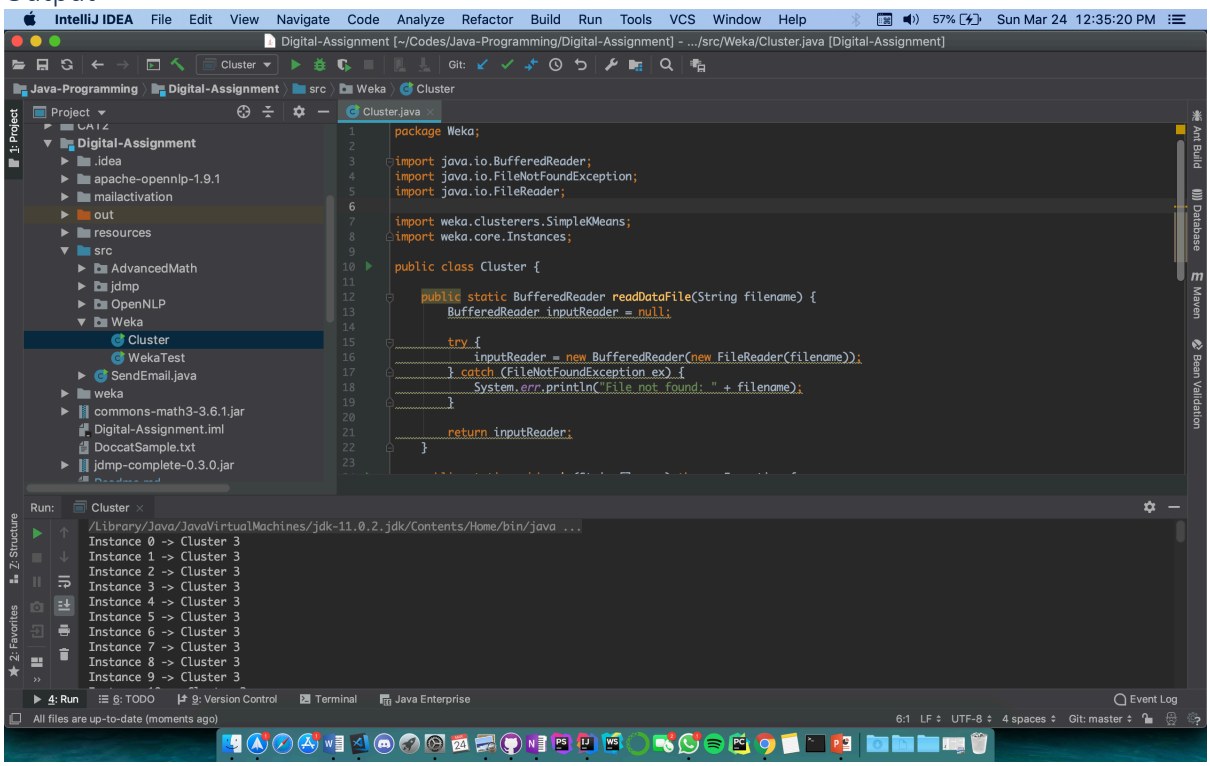
Weka is a collection of machine learning algorithms for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization.

## Code

```
package Weka;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;

import weka.clusterers.SimpleKMeans;
import weka.core.Instances;
```

```java
public class Cluster {

    public static BufferedReader readDataFile(String filename) {
        BufferedReader inputReader = null;

        try {
            inputReader = new BufferedReader(new FileReader(filename));
        } catch (FileNotFoundException ex) {
            System.err.println("File not found: " + filename);
        }

        return inputReader;
    }

    public static void main(String[] args) throws Exception {
        SimpleKMeans kmeans = new SimpleKMeans();

        kmeans.setSeed(10);

        //important parameter to set: preserver order, number of cluster.
        kmeans.setPreserveInstancesOrder(true);
        kmeans.setNumClusters(5);

        BufferedReader datafile = readDataFile("resources/data/iris.arff");
        Instances data = new Instances(datafile);


        kmeans.buildClusterer(data);

        // This array returns the cluster number (starting with 0) for each
instance
        // The array has as many elements as the number of instances
        int[] assignments = kmeans.getAssignments();

        int i=0;
        for(int clusterNum : assignments) {
            System.out.printf("Instance %d -> Cluster %d \n", i, clusterNum);
            i++;
        }
    }
}
```

## Output




## Classification in java

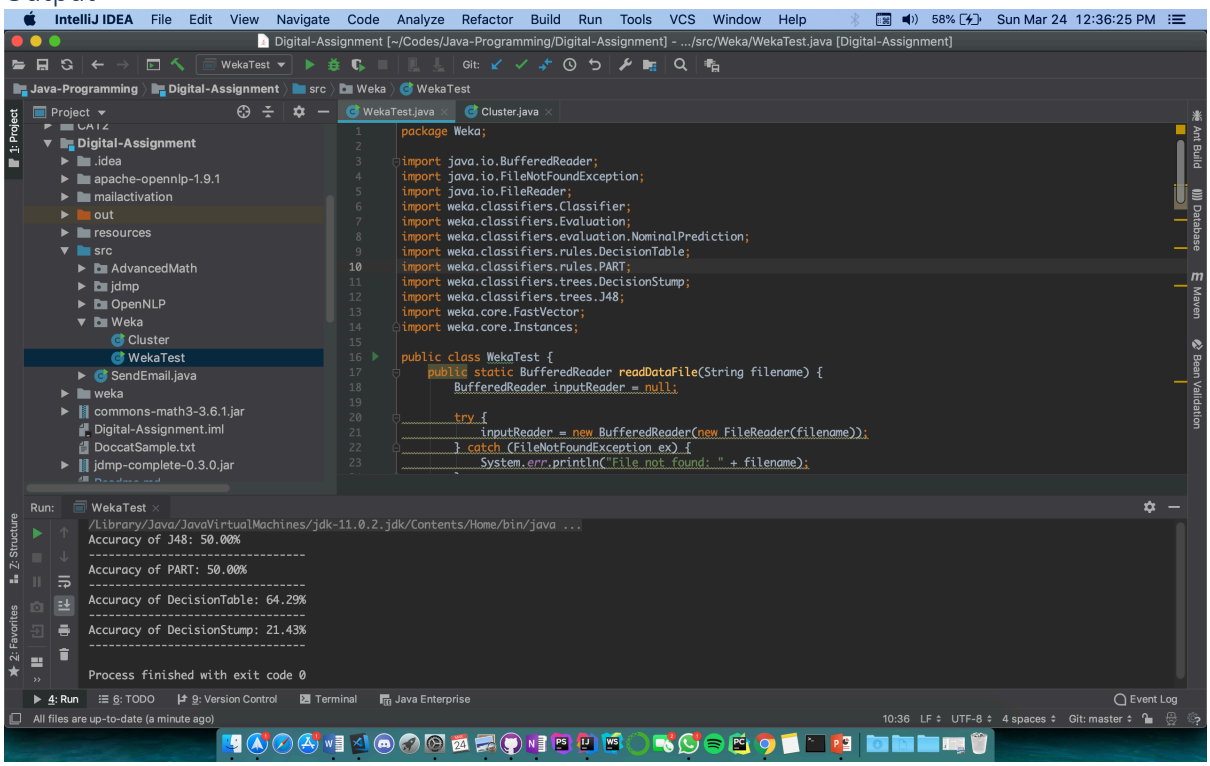
### Code

```java
package Weka;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import weka.classifiers.Classifier;
import weka.classifiers.Evaluation;
import weka.classifiers.evaluation.NominalPrediction;
import weka.classifiers.rules.DecisionTable;
import weka.classifiers.rules.PART;
import weka.classifiers.trees.DecisionStump;
import weka.classifiers.trees.J48;
import weka.core.FastVector;
import weka.core.Instances;

public class WekaTest {
    public static BufferedReader readDataFile(String filename) {
        BufferedReader inputReader = null;

        try {
            inputReader = new BufferedReader(new FileReader(filename));
        } catch (FileNotFoundException ex) {
            System.err.println("File not found: " + filename);
        }

        return inputReader;
    }
```

```
    public static Evaluation classify(Classifier model,
                                      Instances trainingSet, Instances
testingSet) throws Exception {
        Evaluation evaluation = new Evaluation(trainingSet);

        model.buildClassifier(trainingSet);
        evaluation.evaluateModel(model, testingSet);

        return evaluation;
    }

    public static double calculateAccuracy(FastVector predictions) {
        double correct = 0;

        for (int i = 0; i < predictions.size(); i++) {
            NominalPrediction np = (NominalPrediction) predictions.elementAt(i);
            if (np.predicted() == np.actual()) {
                correct++;
            }
        }

        return 100 * correct / predictions.size();
    }

    public static Instances[][] crossValidationSplit(Instances data, int
numberOfFolds) {
        Instances[][] split = new Instances[2][numberOfFolds];

        for (int i = 0; i < numberOfFolds; i++) {
            split[0][i] = data.trainCV(numberOfFolds, i);
            split[1][i] = data.testCV(numberOfFolds, i);
        }

        return split;
    }

    public static void main(String[] args) throws Exception {
        BufferedReader datafile = readDataFile("resources/weather.txt");

        Instances data = new Instances(datafile);
        data.setClassIndex(data.numAttributes() - 1);

        // Do 10-split cross validation
        Instances[][] split = crossValidationSplit(data, 10);

        // Separate split into training and testing arrays
        Instances[] trainingSplits = split[0];
        Instances[] testingSplits = split[1];

        // Use a set of classifiers
        Classifier[] models = {
                new J48(), // a decision tree
                new PART(),
                new DecisionTable(),//decision table majority classifier
                new DecisionStump() //one-level decision tree
        };

        // Run for each model
```

```
        for (int j = 0; j < models.length; j++) {

            // Collect every group of predictions for current model in a
FastVector
            FastVector predictions = new FastVector();

            // For each training-testing split pair, train and test the
classifier
            for (int i = 0; i < trainingSplits.length; i++) {
                Evaluation validation = classify(models[j], trainingSplits[i],
testingSplits[i]);

                predictions.appendElements(validation.predictions());

                // Uncomment to see the summary for each training-testing pair.
                //System.out.println(models[j].toString());
            }

            // Calculate overall accuracy of current classifier on all splits
            double accuracy = calculateAccuracy(predictions);

            // Print current classifier's name and accuracy in a complicated,
            // but nice-looking way.
            System.out.println("Accuracy of " +
models[j].getClass().getSimpleName() + ": "
                    + String.format("%.2f%%", accuracy)
                    + "\n---------------------------------");
        }

    }
}
```

## Output




## Linear Regression Classification in Java

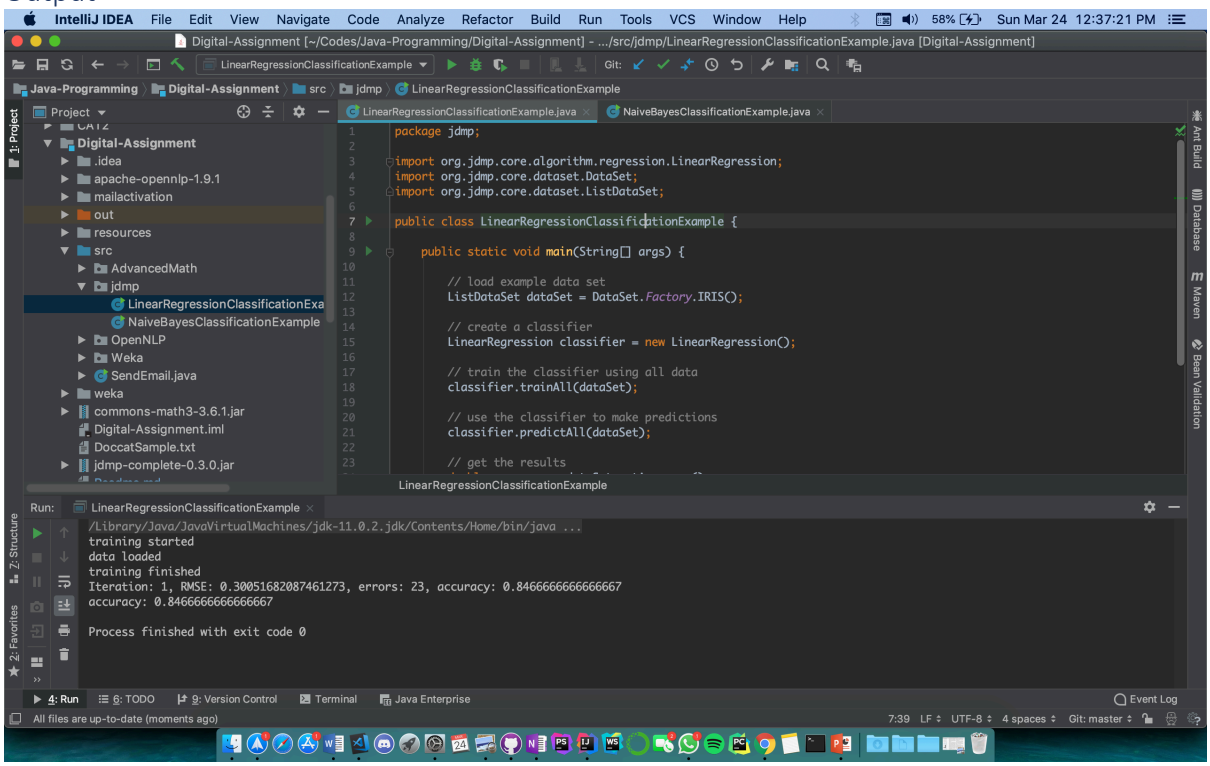
The Java Data Mining Package (JDMP) is an open source Java library for data analysis and machine learning. It facilitates the access to data sources and machine learning algorithms (e.g. clustering, regression, classification, graphical models, optimization) and provides visualization modules. JDMP provides a number of algorithms and tools, but also interfaces to other machine learning and data mining packages (Weka, LibLinear, Elasticsearch, LibSVM, Mallet, Lucene, Octave).

## Code

```
package jdmp;

import org.jdmp.core.algorithm.regression.LinearRegression;
import org.jdmp.core.dataset.DataSet;
import org.jdmp.core.dataset.ListDataSet;

public class LinearRegressionClassificationExample {

    public static void main(String[] args) {

        // load example data set
        ListDataSet dataSet = DataSet.Factory.IRIS();

        // create a classifier
        LinearRegression classifier = new LinearRegression();

        // train the classifier using all data
        classifier.trainAll(dataSet);

        // use the classifier to make predictions
        classifier.predictAll(dataSet);
```

```
        // get the results
        double accurary = dataSet.getAccuracy();

        System.out.println("accuracy: " + accurary);
    }

}
```

## Output




## Naïve Bayes Classification in Java

### Code

```
package jdmp;
import org.jdmp.core.algorithm.classification.bayes.NaiveBayesClassifier;
import org.jdmp.core.dataset.DataSet;
import org.jdmp.core.dataset.ListDataSet;

public class NaiveBayesClassificationExample {

    public static void main(String[] args) {

        // load example data set
        ListDataSet dataSet = DataSet.Factory.IRIS();

        // create a classifier
        NaiveBayesClassifier classifier = new NaiveBayesClassifier();

        // train the classifier using all data
        classifier.trainAll(dataSet);
```

```
        // use the classifier to make predictions
        classifier.predictAll(dataSet);

        // get the results
        double accurary = dataSet.getAccuracy();

        System.out.println("accuracy: " + accurary);
    }
}
```

## Output