# **Programming Assignment #3**

**Program # 3.2: Option2:**

import java.util.Scanner;

public class Assignment3 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Programming Fundamentals");

System.out.println("Program #3");

System.out.println("Name: Ravikumar Naik");

System.out.println("Fall 2021");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

// Loop to keep asking for input until -1 is entered

while (true) {

System.out.println("Please enter a number:");

int number = scanner.nextInt();

if (number == -1) {

System.out.println("Goodbye!");

break;

}

boolean hasOdd = hasAnOddDigit(number);

int sum = sumOfDigits(number, hasOdd);

if (hasOdd) {

System.out.println("True. The number has at least one odd digit. The sum of the odd digits is " + sum);

} else {

System.out.println("False. The number has no odd digits. The sum of the even digits is " + sum);

}

}

scanner.close();

}

public static boolean hasAnOddDigit(int number) {

while (number != 0) {

int digit = number % 10;

if (digit % 2 != 0) {

return true;

}

number /= 10;

}

return false;

}

public static int sumOfDigits(int number, boolean odd) {

int sum = 0;

while (number != 0) {

int digit = number % 10;

if ((odd && digit % 2 != 0) || (!odd && digit % 2 == 0)) {

sum += digit;

}

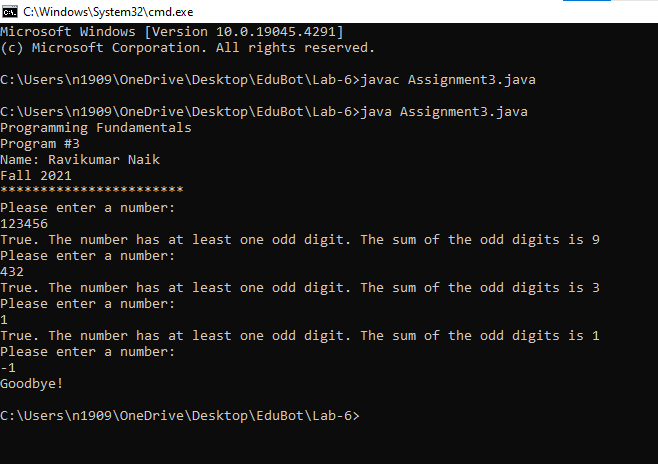
number /= 10;

}

return sum;

}

}



**Programming Assignment 3.1 - Machine Learning**

**import java.io.File;**

**import java.io.FileNotFoundException;**

**import java.util.Scanner;**

**public class NearestNeighbor{**

**public static void main(String[] args) {**

**//inputting file names**

**Scanner scanner = new Scanner(System.in);**

**System.out.print("Enter filename for training dataset: ");**

**String trainingFilename = scanner.nextLine();**

**System.out.print("Enter filename for testing dataset: ");**

**String testingFilename = scanner.nextLine();**

**//Loading and parsing dataset files**

**double[][] trainingAttributes = loadAttributes(trainingFilename);**

**double[][] testingAttributes = loadAttributes(testingFilename);**

**String[] trainingLabels = loadLabels(trainingFilename);**

**String[] testingLabels = loadLabels(testingFilename);**

**// Classify testing examples**

**String[] predictedLabels = classifyExamples(trainingAttributes, trainingLabels, testingAttributes);**

**//For accuracy**

**int matches = 0;**

**for (int i = 0; i < testingLabels.length; i++) {**

**if (testingLabels[i].equals(predictedLabels[i])) {**

**matches++;**

**}**

**}**

**double accuracy = (double) matches / testingLabels.length;**

**// Output results**

**System.out.println("EX#: TRUE LABEL, PREDICTED LABEL");**

**for (int i = 0; i < testingLabels.length; i++) {**

**System.out.println((i + 1) + ": " + testingLabels[i] + " " + predictedLabels[i]);**

**}**

**System.out.println("Accuracy: " + accuracy);**

**}**

**private static double[][] loadAttributes(String filename) {**

**try {**

**Scanner scanner = new Scanner(new File(filename));**

**int numExamples = 75; // Assuming 75 examples**

**double[][] attributes = new double[numExamples][4];**

**// 4 attributes: sepal length, sepal width, petal length, petal width**

**int index = 0;**

**while (scanner.hasNextLine()) {**

**String[] lineValues = scanner.nextLine().split(",");**

**for (int i = 0; i < 4; i++) {**

**attributes[index][i] = Double.parseDouble(lineValues[i]);**

**}**

**index++;**

**}**

**scanner.close();**

**return attributes;**

**} catch (FileNotFoundException e) {**

**System.err.println("File not found: " + filename);**

**return null;**

**} catch (NumberFormatException e) {**

**System.err.println("Error parsing attribute value.");**

**return null;**

**}**

**}**

**private static String[] loadLabels(String filename) {**

**try {**

**Scanner scanner = new Scanner(new File(filename));**

**int numExamples = 75; // Assuming 75 examples**

**String[] labels = new String[numExamples];**

**int index = 0;**

**while (scanner.hasNextLine()) {**

**String[] lineValues = scanner.nextLine().split(",");**

**labels[index] = lineValues[4];**

**//label is in the 5th column**

**index++;**

**}**

**scanner.close();**

**return labels;**

**} catch (FileNotFoundException e) {**

**System.err.println("File not found: " + filename);**

**return null;**

**}**

**}**

**private static String[] classifyExamples(double[][] trainingAttributes, String[] trainingLabels, double[][] testingAttributes) {**

**String[] predictedLabels = new String[testingAttributes.length];**

**for (int i = 0; i < testingAttributes.length; i++) {**

**int closestIndex = findClosest(trainingAttributes, testingAttributes[i]);**

**predictedLabels[i] = trainingLabels[closestIndex];**

**}**

**return predictedLabels;**

**}**

**private static int findClosest(double[][] trainingAttributes, double[] testingAttributes) {**

**double minDistance = Double.MAX\_VALUE;**

**int closestIndex = -1;**

**for (int i = 0; i < trainingAttributes.length; i++) {**

**double distance = calculateDistance(trainingAttributes[i], testingAttributes);**

**if (distance < minDistance) {**

**minDistance = distance;**

**closestIndex = i;**

**}**

**}**

**return closestIndex;**

**}**

**private static double calculateDistance(double[] instance1, double[] instance2) {**

**double sum = 0;**

**for (int i = 0; i < instance1.length; i++) {**

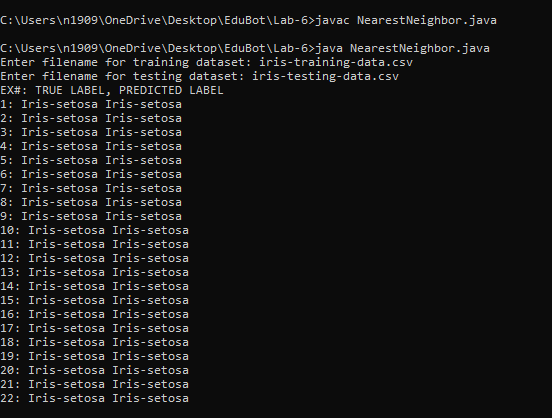
**sum += Math.pow(instance1[i] - instance2[i], 2);**

**}**

**return Math.sqrt(sum);**

**}**

**}**

****