1. 问题提出：

在当前网络社会的交流下，绝大多数的用户认证系统都是通过Username－Password－Verification模式来完成的，这样的方式确实极为有效，但可能会因为某些现实原因，例如在邮箱被盗的时候要求使用验证邮箱、或者手机失窃时通过手机获取验证码等等，导致并不能很好的通过验证码等方式来实现准确验证用户。因此，需要一种高效且准确的方案来实现对用户的精准识别。

而目前来说，已经出现了像人脸识别、虹膜识别、语音识别、指纹识别等等各种已有成效的物理层面上的新兴识别技术。但对于类似于人机交互的时候，纯粹的物理识别已经并不能完全生效。而一种在算法层面上的识别技术：keystroke/typing dynamics （按键/打字习惯分析） 则或许将有奇效。

1. 技术分析：

首先获取某个用户对同一段语句的输入时间间隔（需要多组数据），然后将对各个时间间隔进行同期比对分析。并且通过理论分析认为样本数据满足正态分布，我们可以利用概率论与数理统计的知识，依据样本均值和样本方差去近似估计样本满足的正态分布的𝝁与𝝈，而对新的数据若在（𝝁－𝝈，𝝁＋𝝈）内取权值为10，（𝝁－2𝝈，𝝁＋2𝝈）内权值为5，（𝝁－3𝝈，𝝁＋3𝝈）内权值为1，以外则取0，最后通过求和得出预估值。

1. 代码实现

采用java语言，主要通过GUI界面以及KeyListener 接口来实现输出并记录每个按键的字符、按键持续时长、相邻两次按键之间的时间间隔（分为按下第一次到按下第二次的时间ppTime 与松开第一次到按下第二次的时间prTime）

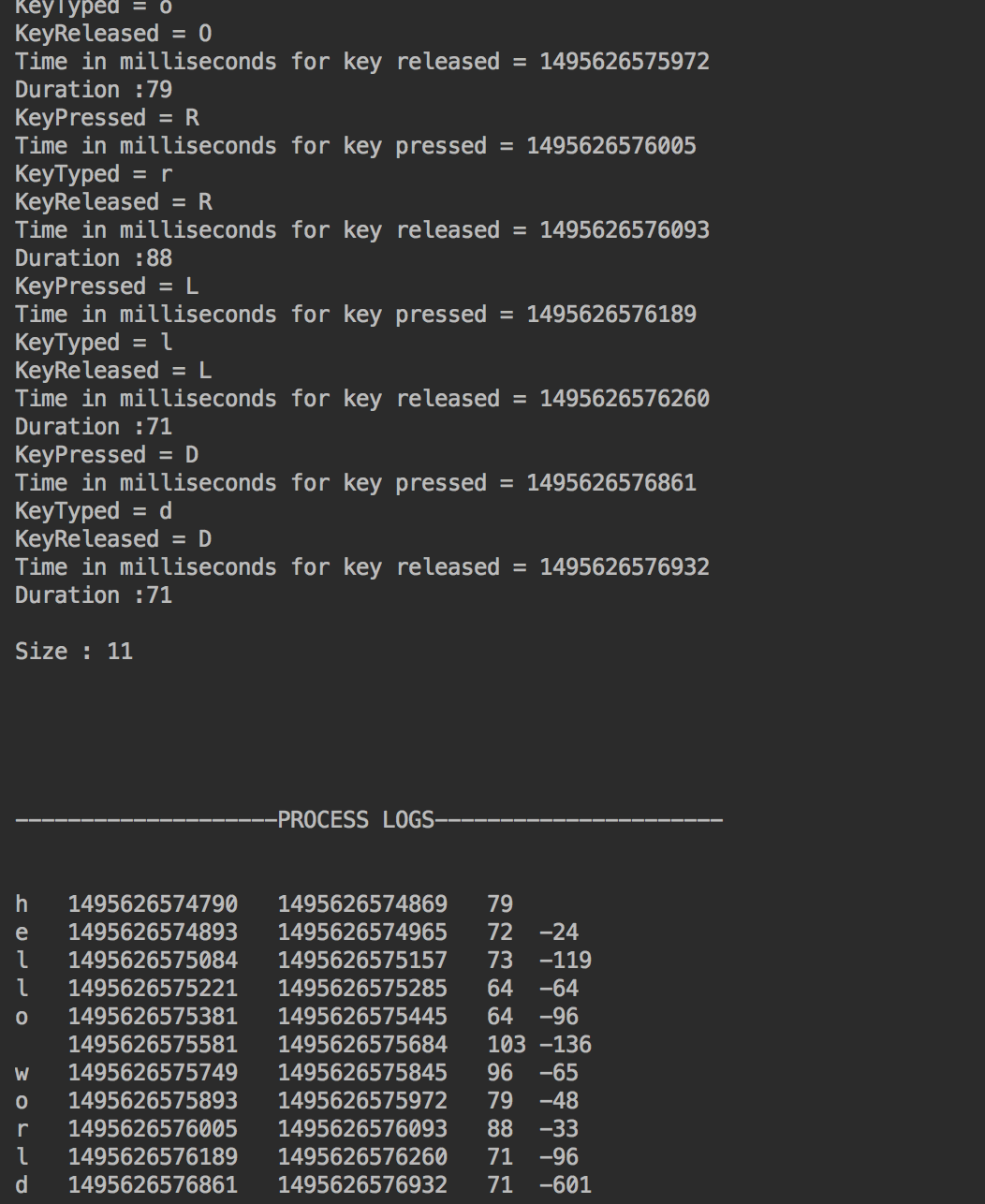
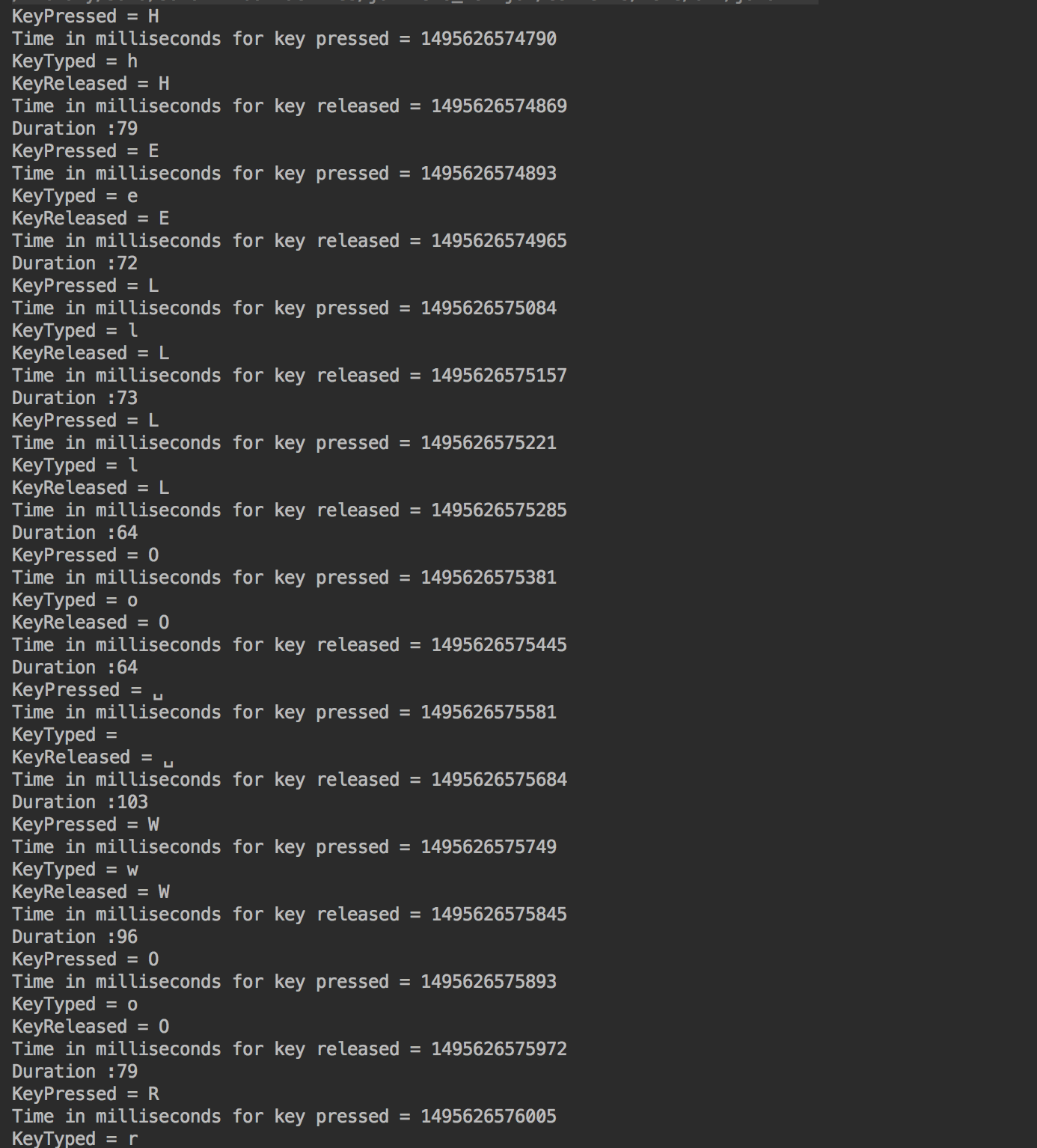
package com;  
 import java.awt.GridLayout;  
 import java.awt.event.KeyListener;  
 import java.awt.event.WindowAdapter;  
 import java.awt.event.WindowEvent;  
 import java.io.File;  
 import java.io.FileWriter;  
 import java.awt.event.ActionEvent;  
 import java.awt.event.ActionListener;  
 import java.awt.event.KeyEvent;  
 import java.util.ArrayList;  
  
 import javax.swing.JButton;  
 import javax.swing.JFrame;  
 import javax.swing.JLabel;  
 import javax.swing.JPanel;  
 import javax.swing.JTextField;  
  
public class KeystrokeLogger extends JFrame implements KeyListener {  
  
 private static final long *serialVersionUID* = 1L;  
  
 private JFrame mainFrame;  
 private JPanel textPanel;  
 private JPanel inputPanel;  
 private JPanel btnPanel;  
 private JLabel textArea;  
 private JTextField jtfInput;  
 private JButton okBtn;  
 private JButton canBtn;  
  
 private long pressTime;  
 private long releaseTime;  
 private long pressDuration;  
  
 private static ArrayList<Character> *keyEvents*;  
 private static ArrayList<Long> *pressEvents*;  
 private static ArrayList<Long> *releaseEvents*;  
 private static ArrayList<Long> *holdTime*;  
 private static ArrayList<Long> *ppTime*;  
 private static ArrayList<Long> *prTime*;  
  
 private static boolean *append*;  
  
 private static final String *TEXT\_MESSAGE* = "Type 'hello world' and Click Okay";  
  
 private static File *textFile*;  
  
 public KeystrokeLogger() { createGui(); }  
  
 private void createGui() { // 创建"KeyStroke Dynamics"GUI  
 try {  
 mainFrame = new JFrame("Keystroke Dynamics");  
 mainFrame.setSize(1000, 200);  
 mainFrame.setLayout(new GridLayout(3, 1));  
  
 mainFrame.addWindowListener(new WindowAdapter() {  
 public void windowClosing(WindowEvent windowEvent) {  
 System.*exit*(0);  
 }  
 });  
  
 textPanel = new JPanel();  
 inputPanel = new JPanel();  
 btnPanel = new JPanel();  
 mainFrame.add(textPanel);  
 mainFrame.add(inputPanel);  
 mainFrame.add(btnPanel);  
  
 textArea = new JLabel(*TEXT\_MESSAGE*);  
 jtfInput = new JTextField(20);  
 jtfInput.addKeyListener(this);  
 textPanel.add(textArea);  
 inputPanel.add(jtfInput);  
  
 okBtn = new JButton("Okay");  
 canBtn = new JButton("Cancel");  
 okBtn.setActionCommand("Okay");  
 canBtn.setActionCommand("Cancel");  
 okBtn.addActionListener(new ButtonClickListener());  
 canBtn.addActionListener(new ButtonClickListener());  
 btnPanel.add(okBtn);  
 btnPanel.add(canBtn);  
  
 mainFrame.setVisible(true);  
 } catch(Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void keyTyped(KeyEvent ke) { // 获得按下的字母并记录到KeyEvents中  
 try {  
 Character ch = new Character(ke.getKeyChar());  
 System.*out*.println("KeyTyped = " + ch.charValue());  
 *keyEvents*.add(ch);  
 } catch(Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void keyPressed(KeyEvent ke) { // 记录按下某一按键的时刻  
 try {  
 Character ch = new Character(KeyEvent.*getKeyText*(ke.getKeyCode()).toCharArray()[0]);  
 System.*out*.println("KeyPressed = " + ch.charValue());  
 System.*out*.print("Time in milliseconds for key pressed = ");  
 pressTime = System.*currentTimeMillis*();  
 System.*out*.println(pressTime);  
  
 *pressEvents*.add(pressTime);  
 } catch(Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void keyReleased(KeyEvent ke) { // 记录松开某一按键的时刻  
 try {  
 Character ch = new Character(KeyEvent.*getKeyText*(ke.getKeyCode()).toCharArray()[0]);  
 System.*out*.println("KeyReleased = " + ch);  
 System.*out*.print("Time in milliseconds for key released = ");  
 releaseTime = System.*currentTimeMillis*();  
 System.*out*.println(releaseTime);  
 pressDuration = releaseTime - pressTime; // 获取持续时间  
 System.*out*.println("Duration :" + pressDuration);  
  
 *releaseEvents*.add(releaseTime);  
 } catch(Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 private class ButtonClickListener implements ActionListener {  
 public void actionPerformed(ActionEvent ae) {  
 try {  
 String command = ae.getActionCommand();  
 if(command.equals("Okay")) {  
 mainFrame.dispose();  
 processLogs();  
 }  
 else if(command.equals("Cancel")) {  
 mainFrame.dispose();  
 }  
 } catch(Exception e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 private void processLogs() { // 记录用户所打字的内容,每个按键的持续时间及连续两次按键之间的时间间隔  
 try {  
 StringBuilder sb = new StringBuilder();  
 //sb.append("Character,PressTime,ReleaseTime,Duration\n");  
  
 String headers = "KeyStroke Dynamics Logs\n\n";  
  
 sb.append(headers);  
  
 StringBuilder sb1 = new StringBuilder();  
  
  
 System.*out*.println("\nSize : " + *keyEvents*.size() + "\n\n");  
 System.*out*.println("\n\n\n--------------------PROCESS LOGS----------------------\n\n");  
  
 for(int i=0; i<*keyEvents*.size(); i++) {  
 System.*out*.print(*keyEvents*.get(i).charValue()+ "\t");  
 System.*out*.print(*pressEvents*.get(i).longValue() + "\t");  
 System.*out*.print(*releaseEvents*.get(i).longValue() + "\t");  
 System.*out*.print(*releaseEvents*.get(i).longValue() - *pressEvents*.get(i).longValue() + "\t"); // 某一次按键的持续时间  
 if (i != 0) {  
 System.*out*.print(*releaseEvents*.get(i - 1).longValue() - *pressEvents*.get(i).longValue()); // 某一次按键与之前一次松开按键的时间间隔  
 }  
 System.*out*.print("\n");  
 *holdTime*.add(new Long(*releaseEvents*.get(i).longValue() - *pressEvents*.get(i).longValue())); // 某一次按键的持续时间  
 }  
  
 for(int i=1; i<*keyEvents*.size(); i++) {  
 *ppTime*.add(new Long(*pressEvents*.get(i).longValue() - *pressEvents*.get(i-1).longValue())); // 某一次按键与之前一次按键的时间间隔  
 *prTime*.add(new Long(*releaseEvents*.get(i-1).longValue() - *pressEvents*.get(i).longValue())); // 某一次按键与之前一次松开按键的时间间隔  
 }  
  
 sb1.append("Content:\nchar\tholdtime\tppTime\tprTime\n" + *keyEvents*.get(0).charValue() + "\t\t" + *holdTime*.get(0).longValue() + "\n");  
  
 for(int i=0; i<*ppTime*.size(); i++) {  
 sb1.append(*keyEvents*.get(i+1).charValue() + "\t\t" + *holdTime*.get(i+1).longValue() + "\t\t\t" + *ppTime*.get(i).longValue() + "\t\t" + *prTime*.get(i).longValue() + "\n");  
 }  
  
 sb1.append("\n");  
  
 *textFile* = new File("keystroke-dynamics.txt");  
 if(!*textFile*.exists()) {  
 *textFile*.createNewFile();  
 *append* = false;  
 } else {  
 *append* = true;  
 }  
 FileWriter fw = new FileWriter(*textFile*, true);  
 if(*append*) {  
 fw.write(sb1.toString());  
 } else {  
 fw.write(sb.toString());  
 fw.write(sb1.toString());  
 }  
 fw.close();  
  
 } catch(Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static void main(String[] args) {  
 try {  
 *keyEvents* = new ArrayList<Character>();  
 *pressEvents* = new ArrayList<Long>();  
 *releaseEvents* = new ArrayList<Long>();  
 *holdTime* = new ArrayList<Long>();  
 *ppTime* = new ArrayList<Long>();  
 *prTime* = new ArrayList<Long>();  
  
 KeystrokeLogger keystrokeLogger = new KeystrokeLogger();  
 keystrokeLogger.pack();  
  
 } catch(Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

判断已知样本与测试数据的偏差值：NormalJudge.java

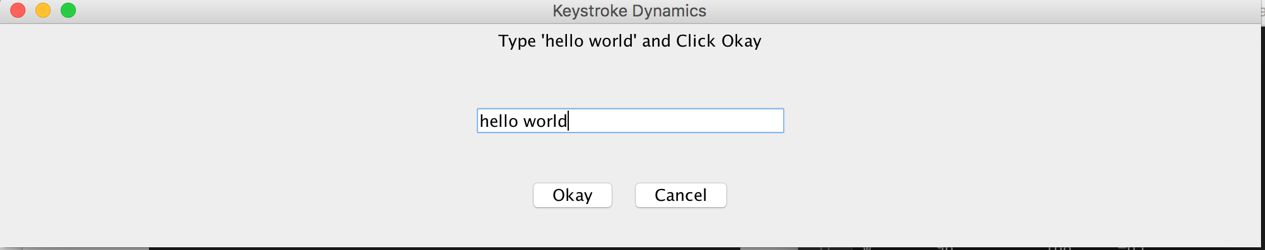
package com;  
  
import java.io.File;  
import java.io.FileReader;  
  
*/\*\*  
 \* Created by liz on 17/5/25.  
 \*/*public class NormalJudge {  
 private static int[][] testArray;  
 private static int[] *newArray*;  
  
 */\*\*  
 \*  
 \** ***@param*** *array 样本数组  
 \** ***@param*** *t 新样本数  
 \** ***@return*** *\*/* private static int judge(int[] array,int t) {  
 double mean = 0; // 样本均值  
 double variance = 0; // 样本方差  
 for(int i = 0; i < array.length; i++) {  
 mean += array[i];  
 System.*out*.print(array[i] + "\t");  
 }  
 System.*out*.println("\n" + t);  
 mean /= array.length;  
  
 for(int i = 0; i < array.length; i++) {  
 variance += array[i]\*array[i];  
 }  
 variance -= array.length \* mean \* mean;  
 variance /= (array.length - 1);  
  
 double standardDeviation = Math.*sqrt*(variance);  
  
 if (t > mean - standardDeviation && t < mean + standardDeviation)  
 return 10; // 若新值在1倍sigma范围内权重10  
 else if (t > mean - 2 \* standardDeviation && t < mean + 2 \* standardDeviation)  
 return 5; // 若新值在2倍sigma范围内权重5  
 else if (t > mean - 3 \* standardDeviation && t < mean + 3 \* standardDeviation)  
 return 1; // 若新值在3倍sigma范围内权重1  
 else  
 return 0; // 若新值在3倍sigma范围外权重0  
  
 }  
  
 public static void main(String[] args) {  
 try {  
 File OriginFile = new File("keystroke-dynamics.txt");  
 File TestFile = new File("KD\_exam.txt");  
 FileReader fileReader1 = new FileReader(OriginFile);  
 char[] chars = new char[3000];  
 fileReader1.read(chars);  
 String[] strings = String.*valueOf*(chars).split("\\s+");  
// for (String string: strings) {  
// System.out.println(string);  
// }  
 int count = 0;  
 testArray = new int[40][10];  
 for (int i = 0; i < 40; i++) {  
 for (int j = 0; j < 10; j++) {  
 testArray[i][j] = Integer.*parseInt*(strings[count++]);  
// System.out.println(testArray[i][j]);  
 }  
 }  
 FileReader fileReader2 = new FileReader(TestFile);  
 char[] chars2 = new char[100];  
 fileReader2.read(chars2);  
 String[] strings2 = String.*valueOf*(chars2).split("\\s+");  
 *newArray* = new int[10];  
 for(int i = 0; i < 10; i++) {  
 *newArray*[i] = Integer.*parseInt*(strings2[i]);  
 }  
  
 int[] tmpArray = new int[40];  
 count = 0;  
 for(int i = 0; i < 10; i++) {  
 for (int j = 0; j < 40; j++) {  
 tmpArray[j] = testArray[j][i];  
 }  
 count += *judge*(tmpArray,*newArray*[i]);  
 }  
  
 System.*out*.println(count);  
 }  
 catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
}

1. 输出界面与结果

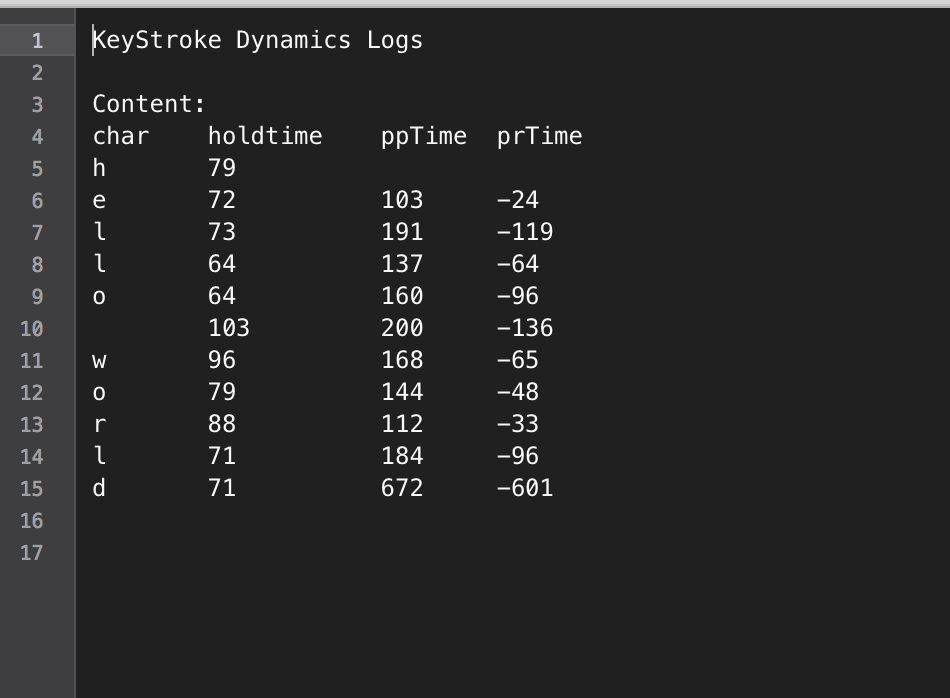
终端输出结果：



GUI界面：



保存到keystroke-dynamics.txt界面结果



通过测试50组数据的样本，进行比对分析：

样本数据：

103 96 128 208 376 176 135 104 113 139

88 128 144 264 168 296 128 96 128 120

71 128 144 248 176 152 120 104 120 136

96 120 128 168 136 216 112 112 112 152

111 113 128 191 177 151 88 105 144 119

87 120 128 160 232 136 96 104 152 112

79 104 128 168 168 136 128 128 152 144

103 112 136 167 169 143 97 95 129 112

79 128 120 160 168 136 152 96 296 144

71 104 120 160 136 152 104 104 120 120

88 112 128 160 160 120 135 129 128 104

103 120 120 176 152 160 112 96 120 128

88 128 120 168 168 136 128 128 160 112

103 120 128 160 160 128 96 112 120 120

103 152 136 176 160 144 88 120 104 128

70 129 119 177 152 160 88 104 120 104

87 96 120 192 160 168 152 208 256 144

96 104 136 224 176 152 112 95 137 96

64 103 128 200 144 128 128 40 200 56

95 112 120 192 216 128 144 144 128 152

96 104 128 176 184 152 88 136 96 152

104 96 111 169 175 144 65 144 96 168

79 88 136 184 152 112 88 136 88 160

95 88 128 160 152 152 48 184 72 200

120 79 113 168 352 104 103 136 80 153

71 88 128 168 152 160 72 136 56 144

106 72 136 144 128 168 48 168 40 168

95 88 120 168 144 144 64 136 72 144

64 87 137 176 144 136 104 104 96 104

47 104 120 176 152 176 208 152 120 120

71 88 120 192 176 152 72 144 64 160

111 89 127 145 168 175 88 137 72 144

55 112 120 176 144 152 88 128 88 128

79 88 128 160 152 152 88 112 96 120

80 96 128 183 177 144 103 105 104 111

87 96 144 176 176 128 104 104 112 112

96 72 112 168 144 184 96 128 104 112

79 88 128 176 160 144 143 89 152 56

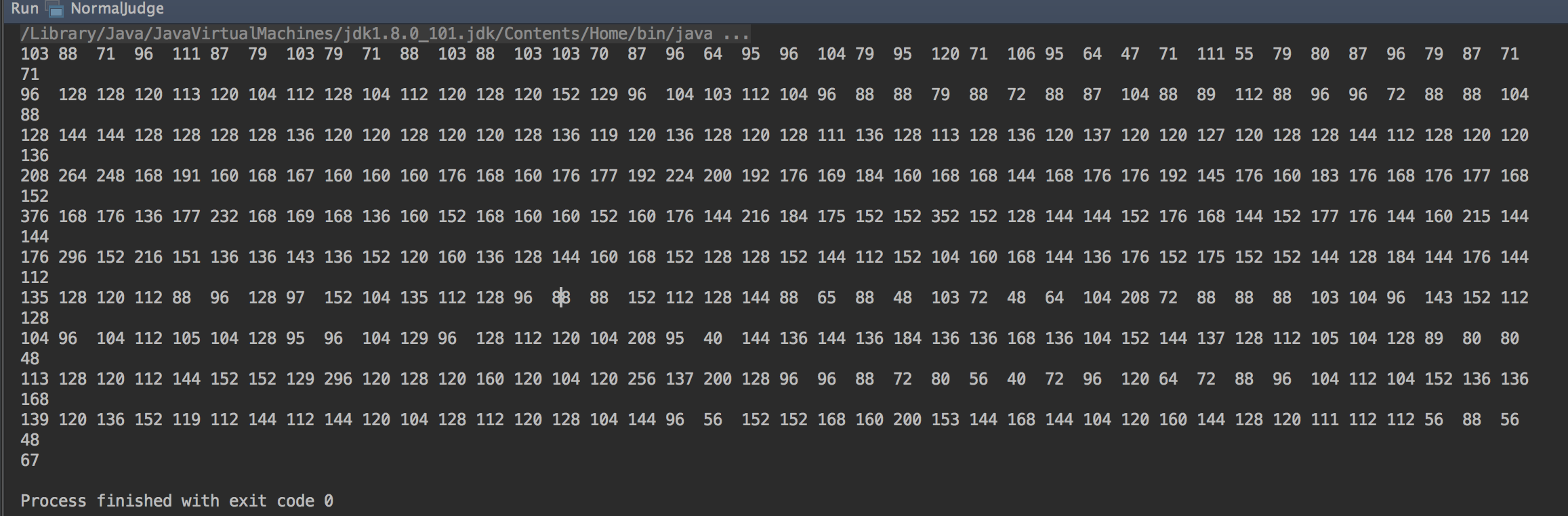
87 88 120 177 215 176 152 80 136 88

71 104 120 168 144 144 112 80 136 56

新测试数据：

71 88 136 152 144 112 128 48 168 48

最后分析结果



得分：67分

1. 仍留有的尚未修复的BUG以及待改善的部分
2. 软件使用中如果使用了输入法可能会对最终结果造成影响（出现异常或holdtime变成负值）
3. 目前尚未限制用户所需要输入的内容（仅仅使用提示语句要求输入hello world，若输入其他语句也不会出现任何错误提示）
4. 在使用过程中如果出现输入错误且使用了退格也会一并记录下来，但似乎会对最后结果产生尚未明确的影响。
5. 对该问题的进一步思考

以下引用自专利

周福才 一种基于击键特征识别的认证方法和系统:中国，CN 101478401 A[P].2009-1-21

