MIDI Humanizer: Final Project Report

Team La-a

Fall 2011

Contents

1 Requirements

Software Requirements Specification

for

MIDI Humanizer

Version 1.0

**Prepared by Jacob Peck,
Jason Shaffner,
and Jamie White**

**State University of New York
College at Oswego**

September, 2011

Table of Contents

1. Introduction

1.1 Purpose

MIDI Humanizer exists to fill a niche in the MIDI representation of performance aspects of a musical piece. The program reads in a MIDI file specified by the user, and applies profiled variance to various dimensions of MIDI voices, such as volume and timing. MIDI Humanizer attempts, in a few words, to humanize a MIDI sequence.

1.2 Document Conventions

When reading this document, it can be assumed that any `monospaced` characters are either direct input from a user, or output from an operation.

1.3 Intended Audience and Reading Suggestions

This document is prepared for developers, project managers, users, and documentation writers. Those who wish to read this document are suggested to approach it in a linear order, after first glancing at the section headings as outlined in the table of contents. Of particular note is section 3, describing all of the vital requirements from a developer's perspective.

1.4 Project Scope

MIDI Humanizer is an attempt to reintroduce the human element to MIDI streams, particularly those that have been quantized. This will allow computer musicians to simulate human imperfections. This project seeks to offer computer and MIDI musicians an option to increase variance in a simplistic way. In terms of marketability, the potential for this product is low in the general market, but rather high in the digital audio workstation market.

1.5 References

This document refers to the General MIDI specification, which is available online at <http://www.midi.org/techspecs/index.php>, as well as the JFugue User's Guide, which is available for purchase for $25 from <http://www.jfugue.org/>. The initial project description provided by the client is also referred to, along with any references that the client may provide.

2. Overall Description

2.1 Product Perspective

MIDI Humanizer is a stand-alone project operating in a field with little competition. MIDI Humanizer does not seek to replace any available projects, nor does it seek to add on to any existing products.

2.2 Product Features

REQ22-1: MIDI Humanizer will offer a simple graphical user interface.
REQ22-2: The interface will allow the user to import files to be worked upon, select variable levels of variance and channels to operate on, provide an output file, and initiate the process.
REQ22-3: MIDI Humanizer will allow for previews during every step of the way.
REQ22-4: The variance applied to the sequence will be determined by a user-selectable (and user-definable) profile, allowing for emulating different styles of playing.

2.3 User Classes and Characteristics

Computer Musicians and Mixing Technicians – These users have a decent level of familiarity with MIDI, and as such should find no problem using this software. Advanced options such as profiling will be geared towards these users.
Amateur Musicians – These users may not know much about the MIDI spec, but they are able to click buttons and receive results. Advanced options will be hidden from them if they choose.
MIDI researchers – These users know the MIDI spec inside and out, and may find that this program doesn't suit their needs.

2.4 Operating Environment

REQ24-1: MIDI Humanizer will be written in Java, and will run anywhere where there is a JDK available that implements a standards-compliant MIDI device. OpenJDK will work on several platforms (excluding ARM).
REQ24-2: The preferred platform will be the official OracleJDK, which has the widest coverage of standards-compliant MIDI device implementation. Any version of the Oracle JDK from Java 6 and beyond will work, though preferential treatment will be given towards the optimizations present in Java 7.

2.5 Design and Implementation Constraints

REQ25-1: Source code will be optimized for maintainability and reusability.
REQ25-2: Source will be documented with comments (make "bus code").
REQ25-3: The project will stick solely to the MIDI interface.
REQ25-4: All programmers will be expected to use javadoc to document their methods and classes.

2.6 User Documentation

REQ26-1: On-line help will be available within the program at the user's request.
REQ26-2: The interface will be intuitive.
REQ26-3: Help will be developed as the GUI develops.

2.7 Assumptions and Dependencies

This project assumes that MIDI and JFugue will work as advertised.

3. System Features

3.1 Input/Output

3.1.1 Description and Priority

This feature has a high priority. This feature will read in and send back out a MIDI file. The benefit of having a I/O stream is that it can read in most kinds of files already we just need to do minor adjustments so it can read in a MIDI file. Penalties of this is that we have to find a way to make it platform independent so it could work not only on Windows but on Mac OS and Linux as well.

3.1.2 Stimulus/Response Sequences

The user the user will request a file be loaded. This will be handled by the file I/O method. A message will show up confirming that the file was received and it was in the right format or that it was not received or it was a null file. A pointer will be given no how to fix the mistake but ultimately it is up to the user.

3.1.3 Functional Requirements

REQ31-1: In Order to use this feature this user must have a MIDI file as a input and java must be on the machine. The method will respond to reading in the file and use a series of exceptions to catch a error in the file if the file is of poor quality and tell you the length of the file.
REQ31-2: Should the user try to send in a invalid file type the user should get a message telling them that the file type was invalid and give them a clue to the proper extension.
REQ31-3: This Function should read in the file to be edited and Send back the file when this is complete. This method will also try to catch errors in the file and give the user a warning when a error is found and a easy way to fix it.

3.2 Algorithmic Manipulation of Random Input Sources

3.2.1 Description and Priority

This feature has a high priority. The algorithms must analyze the MIDI sequence, and manipulate them to provide a level of randomness to both amplitude and timing of individual note events. The randomness should move along a bell curve to provide most-probable simulation of said characteristics, while normalizing subsequent notes, as to provide consistency in tempo.

3.2.2 Stimulus/Response Sequences

The user will be able to select “humanness” profiles at various levels, such as “beginner” which will simulate a radical degree of imperfection (along a wider bell curve), or “expert” which will allow for finer imperfections (along a narrower bell curve).

3.2.3 Functional Requirements

REQ-32-1: The algorithms should be applicable to a wide range of inputs, such as different instrumentations, multiple channels, different tempos, etc.
REQ-32-2: The Box Cox Transformation will be applied to create random numbers over a standard normal distribution rather than a uniform distribution.

REQ-32-3: A Gaussian algorithm will be provided in addition to the Box Cox Transform.

3.3 MIDI Sequence Manipulation

3.3.1 Description and Priority

This feature has a high priority, and is critical for the delivery of a finished product. This module of the software will comprise taking information provided by the above two modules and manipulating it so as to add a more human "feel" to it, before sending it back out to the I/O module for writes to disk and sonification if desired.

3.3.2 Stimulus/Response Sequences

This action is performed upon the user confirming via the user-interface that humanization should take place. Humanization happens in the background, and is a non-interactive process, after determinations such as profile, parameters, and input have been determined and confirmed.

3.3.3 Functional Requirements

REQ-33-1: The process must be invisible to the user, to provide a sane user experience.
REQ-33-2: The process must have selectable parameters of the input stream and work upon them individually (i.e. volume, timing, etc.), to allow the user to apply humanization to only certain factors.
REQ-33-3: The process must supply the algorithmic random number generator with a user-selected profile curve.
REQ-33-4: The process must happen in memory and only be committed to disk upon user confirmation.
REQ-33-5: The process must be user-previewable before confirmation to save.
REQ-33-6: The I/O subsystem and algorithmic random number generator must be loosely coupled, to ease maintainability (i.e. use standard JDK and JFugue classes and encapsulate behavior).
REQ-33-7: The process must ensure that the length of the input and output are consistent lengths within a certain tolerance value.

4. External Interface Requirements

4.1 User Interfaces

REQ41-1: There will be a graphical user interface with a unified design.
REQ41-2: A tool bar at the top of the frame will serve as the main method of navigation.
REQ41-3: Dialog boxes will be used to confirm user actions.
REQ41-4: There will be a help option.
REQ41-5: Standard Swing components will be used to create the interface.

4.2 Hardware Interfaces

REQ42-1: Standard input/output devices (monitor, keyboard, mouse, speakers) will be used.

4.3 Software Interfaces

REQ43-1: MIDI Humanizer will make use of the Java platform, and also the JFugue library, version 4 or higher. The Java platform runs on the Java Virtual Machine (JVM), and is available on many platforms.
REQ43-2: The JVM will communicate with the Operating System in order to read in and write out files and communicate with hardware.
REQ43-3: JFugue will communicate with the objects presented by the JVM representing the data on disk. Such objects include representations of notes, sequences, and MIDI files themselves.

4.4 Communications Interfaces

REQ44-1: The only communications handled by MIDI Humanizer is producing sound through JFugue and writing files to disk through the JDK's interface with the OS.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

REQ51-1: The program should not "stall" for longer than 5 seconds when not performing a computation.
REQ51-2: Transformations upon MIDI sequences should not take longer than 5 seconds per minute of MIDI events.
REQ51-3: The interface should be lightweight, such that the controls do not get in the way of user interactions.
REQ51-4: During long periods of computation, a progress bar should be displayed.

5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

5.3 Security Requirements

There are no such concerns with this product.

5.4 Software Quality Attributes

REQ54-1: Should be cross-platform.
REQ54-2: Should be relatively light on system resources. 1Gb ram as a maximum (not due to software bloat, but to allow overhead for JDK), and around 1.2Ghz processing speed.
REQ54-3: Should confirm any "dangerous" user actions, such as file overwrites.
REQ54-4: Should preview MIDI sequences smoothly without hiccups.

6. Other Requirements

N/A.

Appendix A: Glossary

Bus Code: Code so well documented that you could be run over by a bus tomorrow and someone else would be able to pick up where you left off with no problem.

Humanization: The process of applying variance to a quantized MIDI sequence.

JFugue: A Java library for manipulating music. Includes several routines for manipulating MIDI data.

JDK: Java Developer Kit. A collection of Java classes and tools designed to ease Java development.

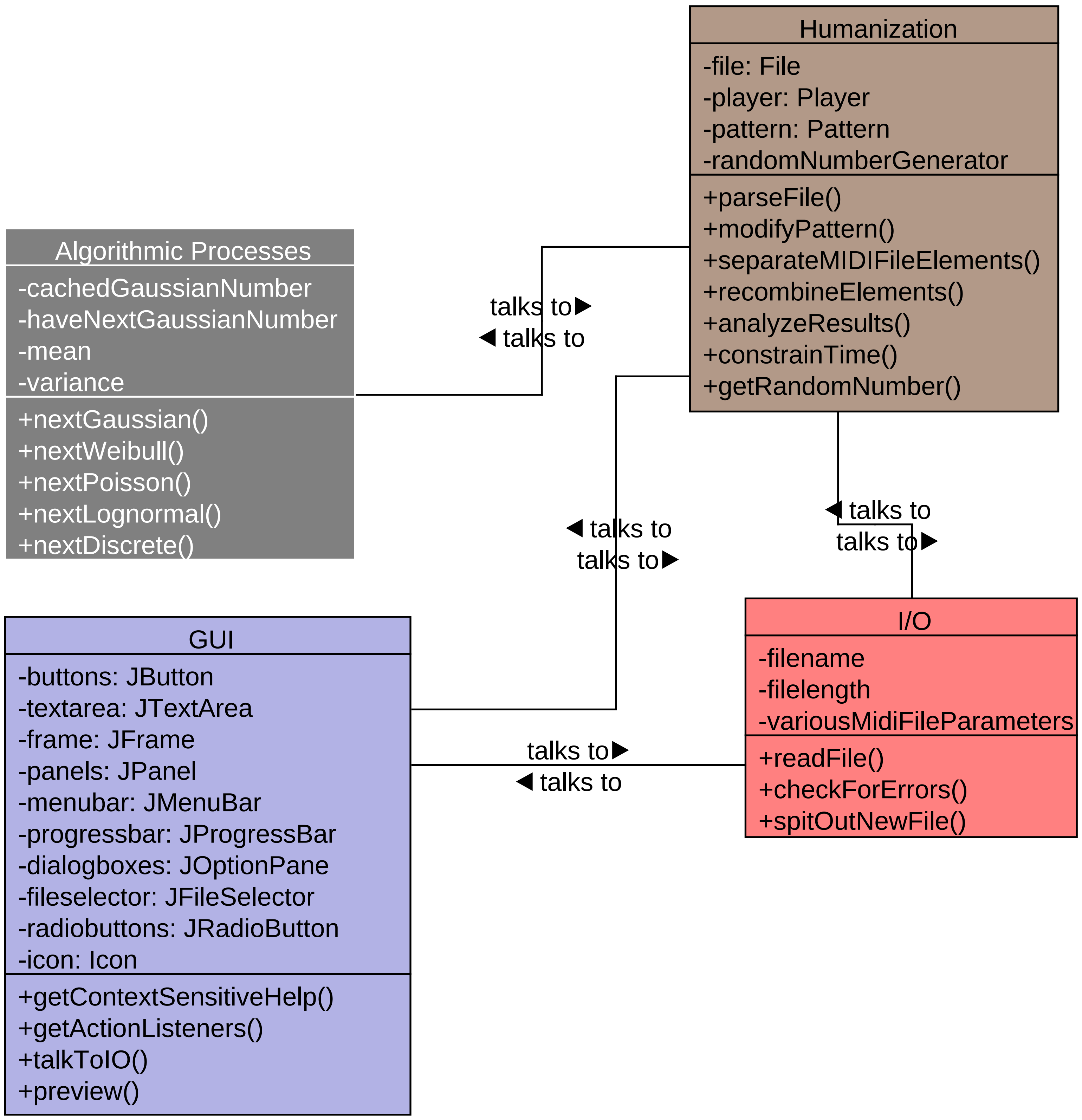
JVM: Java Virtual Machine. The artificial platform that comprises the core of Java technology.

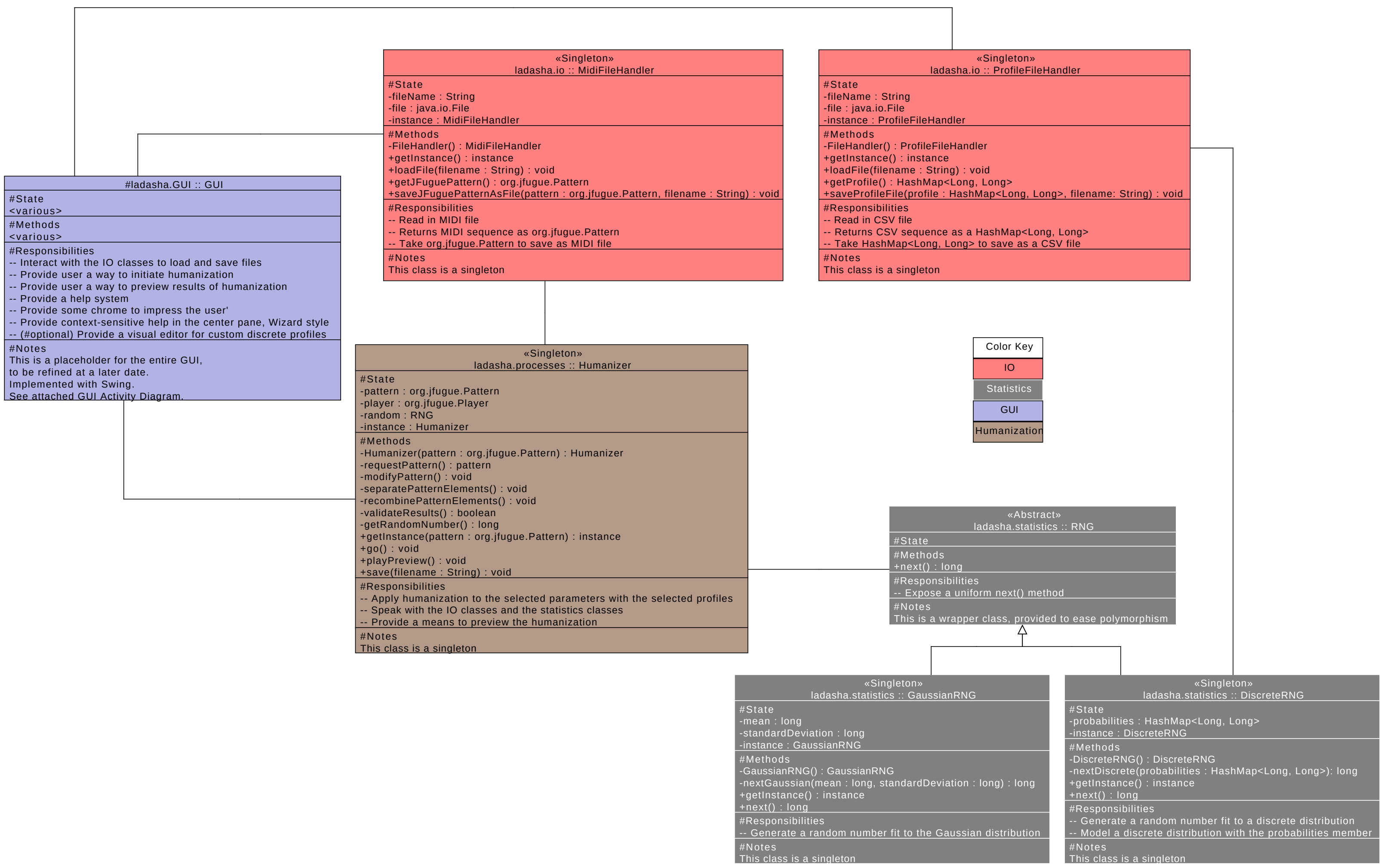
MIDI: Musical Instrument Digital Interface. An industry standard protocol for the exchange and storage of musical data.

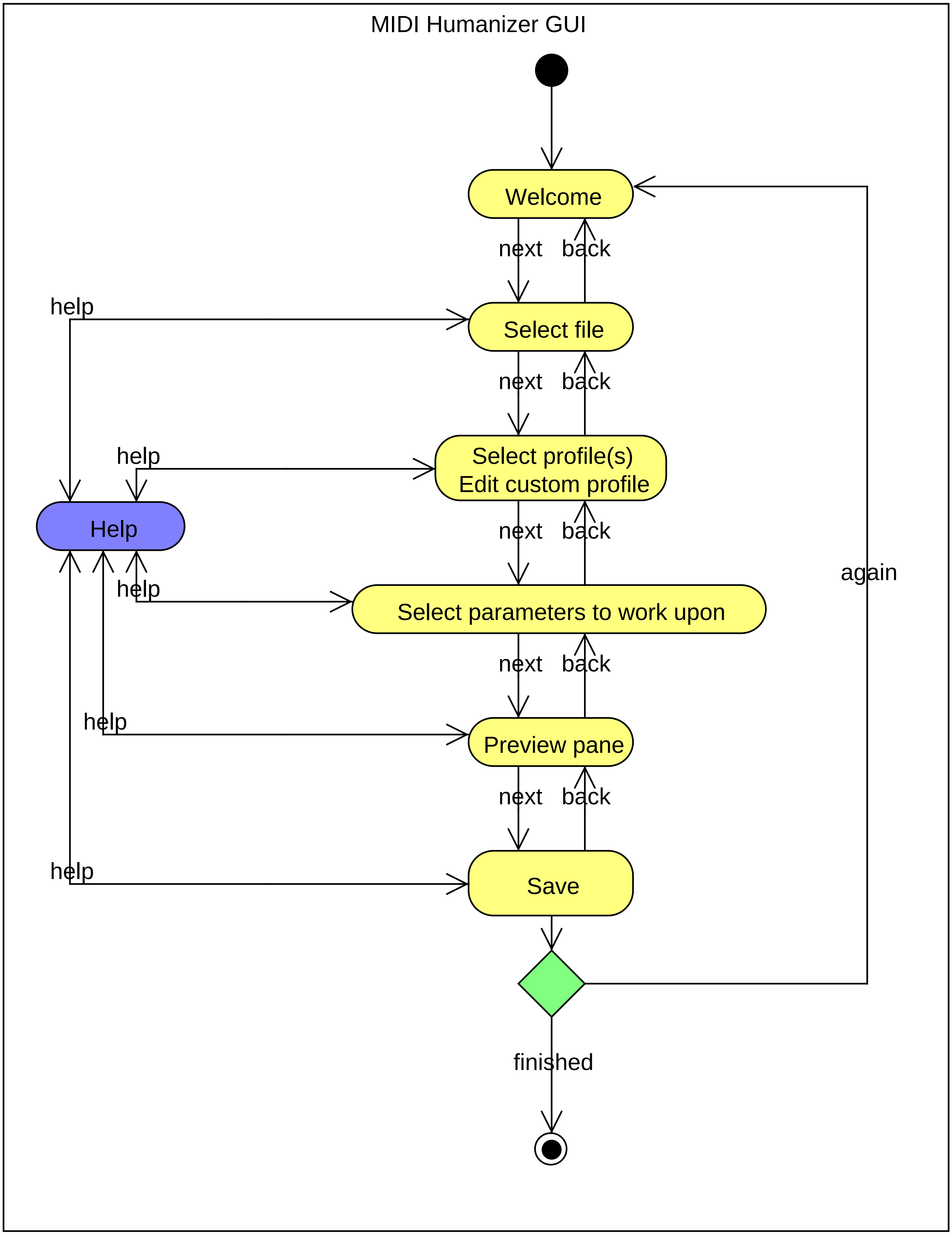
Profile curve: A probability distribution function (pdf) that models the probabilities of variance from some standard average (mean). A good example is the Standard Normal Distribution curve (also called the “bell curve”).

Quantized: A MIDI sequence that has been quantized has been crafted in such a way as to discourage variance, in that all note-on and note-off events are rounded to the nearest multiple of a duration, say the nearest quarter-note.

2 Design


Humanization
-file: File
-player: Player
-pattern: Pattern
-randomNumberGenerator
+parseFile()
+modifyPattern()
+separateMIDIFileElements()
+recombineElements()
+analyzeResults()
+constrainTime()
+getRandomNumber()
Algorithmic Processes
-cachedGaussianNumber
-haveNextGaussianNumber
-mean
-variance
+nextGaussian()
+nextWeibull()
+nextPoisson()
+nextLognormal()
+nextDiscrete()
talks to ▶
◀ talks to
◀ talks to
talks to ▶
◀ talks to
talks to ▶
GUI
-buttons: JButton
-textarea: JTextArea
-frame: JFrame
-panels: JPanel
-menubar: JMenuBar
-progressbar: JProgressBar
-dialogboxes: JOptionPane
-fileselector: JFileSelector
-radiobuttons: JRadioButton
-icon: Icon
+getContextSensitiveHelp()
+getActionListeners()
+talkToIO()
+preview()
I/O
-filename
-filelength
-variousMidiFileParameters
+readFile()
+checkForErrors()
+spitOutNewFile()
talks to ▶
◀ talks to


«Singleton»
ladasha.io :: MidiFileHandler
#State
-fileName : String
-file : java.io.File
-instance : MidiFileHandler
#Methods
-FileHandler() : MidiFileHandler
+getInstance() : instance
+loadFile(filename : String) : void
+getJFuguePattern() : org.jfugue.Pattern
+saveJFuguePatternAsFile(pattern : org.jfugue.Pattern, filename : String) : void
#Responsibilities
-- Read in MIDI file
-- Returns MIDI sequence as org.jfugue.Pattern
-- Take org.jfugue.Pattern to save as MIDI file
#Notes
This class is a singleton
«Singleton»
ladasha.io :: ProfileFileHandler
#State
-fileName : String
-file : java.io.File
-instance : ProfileFileHandler
#Methods
-FileHandler() : ProfileFileHandler
+getInstance() : instance
+loadFile(filename : String) : void
+getProfile() : HashMap<Long, Long>
+saveProfileFile(profile : HashMap<Long, Long>, filename: String) : void
#Responsibilities
-- Read in CSV file
-- Returns CSV sequence as a HashMap<Long, Long>
-- Take HashMap<Long, Long> to save as a CSV file
#Notes
This class is a singleton
#ladasha.GUI :: GUI
#State
<various>
#Methods
<various>
#Responsibilities
-- Interact with the IO classes to load and save files
-- Provide user a way to initiate humanization
-- Provide user a way to preview results of humanization
-- Provide a help system
-- Provide some chrome to impress the user'
-- Provide context-sensitive help in the center pane, Wizard style
-- (#optional) Provide a visual editor for custom discrete profiles
#Notes
This is a placeholder for the entire GUI,
to be refined at a later date.
Implemented with Swing.
See attached GUI Activity Diagram.
«Singleton»
ladasha.processes :: Humanizer
#State
-pattern : org.jfugue.Pattern
-player : org.jfugue.Player
-random : RNG
-instance : Humanizer
#Methods
-Humanizer(pattern : org.jfugue.Pattern) : Humanizer
-requestPattern() : pattern
-modifyPattern() : void
-separatePatternElements() : void
-recombinePatternElements() : void
-validateResults() : boolean
-getRandomNumber() : long
+getInstance(pattern : org.jfugue.Pattern) : instance
+go() : void
+playPreview() : void
+save(filename : String) : void
#Responsibilities
-- Apply humanization to the selected parameters with the selected profiles
-- Speak with the IO classes and the statistics classes
-- Provide a means to preview the humanization
#Notes
This class is a singleton
Color Key
IO
Statistics
GUI
Humanization
«Abstract»
ladasha.statistics :: RNG
#State
#Methods
+next() : long
#Responsibilities
-- Expose a uniform next() method
#Notes
This is a wrapper class, provided to ease polymorphism
«Singleton»
ladasha.statistics :: GaussianRNG
#State
-mean : long
-standardDeviation : long
-instance : GaussianRNG
#Methods
-GaussianRNG() : GaussianRNG
-nextGaussian(mean : long, standardDeviation : long) : long
+getInstance() : instance
+next() : long
#Responsibilities
-- Generate a random number fit to the Gaussian distribution
#Notes
This class is a singleton
«Singleton»
ladasha.statistics :: DiscreteRNG
#State
-probabilities : HashMap<Long, Long>
-instance : DiscreteRNG
#Methods
-DiscreteRNG() : DiscreteRNG
-nextDiscrete(probabilities : HashMap<Long, Long>): long
+getInstance() : instance
+next() : long
#Responsibilities
-- Generate a random number fit to a discrete distribution
-- Model a discrete distribution with the probabilities member
#Notes
This class is a singleton


MIDI Humanizer GUI
Welcome
next
back
Select file
help
next
back
Select profile(s)
Edit custom profile
help
Help
next
back
help
Select parameters to work upon
next
back
help
Preview pane
next
back
help
Save
again
finished

3 Build Plan

Team La-a Build Plan: Alpha

Week of 10/31

- Demo: build alpha1 (GUI Skeleton, disjunct I/O, kerplunk'd algorithms)
- Next build (build alpha2):
  - Glue the I/O into the GUI
  - Flesh out the GUI
  - Work on design of GUI (separate into different tasks per team member)

Week of 11/7

- Demo: build alpha2 (Updated GUI)
- Next build (build alpha3):
  - Implementing profiles
  - Add profile selection to the GUI

Week of 11/14

- Demo: build alpha3 (Updated GUI...again)
- Next build (build alpha4):
  - Implement parameter selection into GUI
  - Implement parameter manipulation code (...pain)

Week of 11/21 (Thanksgiving break, mostly)

- Demo: build alpha4 (Updated GUI)
- Next build (build beta1):
  - Finish implementing parameters
  - Start preview pane functionality

Week of 11/28

- Demo: build beta1 (codename: bravo) (Updated GUI)
- Next build (build beta2):
  - Iron out bugs (run through FindBugs)
  - If time allows, add in custom profile editing feature

Week of 12/5

- Demo: build beta2 (hopefully finished)

4 Test Plans

MIDI Humanizer Test Plan

Team La-a
CSC 380 - Early

Fall 2011

Contents

1 Build System Test Cases

1.1 Compile Code

Importance

Level 1.

Related Documentation

N/A.

Preconditions

- Source code and associated build scripts exist in the current working directory.
- A bash shell is available.

Procedures

1. Run the command `./build.sh`.

Postconditions

- Compiled classes should reside in the `classes/` directory.

Results

Works exactly as expected.

1.2 Run Code

Importance

Level 1.

Related Documentation

N/A.

Preconditions

- Source code and associated build scripts exist in the current working directory.
- A bash shell is available.
- Test 1.1 passed.

Procedures

1. Compile code as in Test 1.1.
2. Run the command `./run.sh`.

Postconditions

- The GUI runs.
- Console input blocks.

Results

Works exactly as expected.

1.3 Make Tarball

Importance

Level 3.

Related Documentation

The unix `tar` `man` pages.

Preconditions

- Source code and associated build scripts exist in the current working directory.
- A bash shell is available.

Procedures

1. Run the command `./maketar.sh`.

Postconditions

- A gzipped tarball containing the contents of the working directory now exists in the working directory.

Results

Works exactly as expected.

2 GUI Test Cases

2.1 GUI Runs

Importance

Level 1.

Related Documentation

N/A.

Preconditions

- Source code and associated build scripts exist in the current working directory.
- A bash shell is available.

Procedures

1. Run the command `./build.sh; ./run.sh`.

Postconditions

- The GUI should run, displaying the welcome screen.

Results

Works exactly as expected.

2.2 Navigation buttons: next

Importance

Level 2.

Related Documentation

N/A.

Preconditions

- Test 2.1 passed.

Procedures

1. Run the command `./build.sh; ./run.sh`.
2. Click the next button to navigate through the panes.

Postconditions

- The GUI should run, and the next pane in the logical sequence should display upon pressing next. The button should be disabled and invisible when there are no more panes.

Results

Works exactly as expected.

2.3 Navigation buttons: previous

Importance

Level 2.

Related Documentation

N/A.

Preconditions

- Test 2.1 passed.
- Test 2.2 passed.

Procedures

1. Run the command `./build.sh; ./run.sh`.
2. Click the next button to navigate through the panes.
3. Click the previous button to navigate backwards.

Postconditions

- The GUI should run, and the previous pane in the logical sequence should display upon pressing previous. The button should be disabled and invisible when there are no more panes.

Results

Works exactly as expected.

2.4 Navigation buttons: help

Importance

Level 2.

Related Documentation

N/A.

Preconditions

- Test 2.1 passed.
- Test 2.2 passed.
- Test 2.3 passed.

Procedures

1. Run the command `./build.sh; ./run.sh`.
2. Use the next and previous buttons to navigate through the panes.
3. Click on the help button on each pane.

Postconditions

- The GUI should run, and upon pressing help, the help screen for each pane should display. During this, the previous button should change text to indicate closing the help screen, and the other navigation buttons (help and next) should be disabled and invisible.

Results

Works exactly as expected.

2.5 MIDI File Loading

Importance

Level 1.

Related Documentation

N/A.

Preconditions

- Test 2.1 passed.
- Test 2.2 passed.
- The user has a MIDI file.

Procedures

1. Run the command `./build.sh; ./run.sh`.
2. Click the next button to navigate through the panes.
3. On the file selection pane, click the select a file button.
4. In the dialog box, select a MIDI file with extension `.mid` or `.midi`.
5. Click the play button to preview.
6. Click the stop button to stop the preview.

Postconditions

- Upon selecting a file to load, the label should change from <no file selected> to the full pathname of the selected file. Clicking the play button at this point will allow the file to be previewed. While previewing a file, the play button should change to a stop button. Clicking this should stop the preview.

Results

Works exactly as expected.

2.6 Profile File Loading

Importance

Level 1.

Related Documentation

N/A.

Preconditions

- Test 2.1 passed.
- Test 2.2 passed.
- The user has several profile files.

Procedures

1. Run the command `./build.sh; ./run.sh`.
2. Click the next button to navigate through the panes.
3. On the profile selection pane, click the add profile button.
4. In the dialog box, select a profile file with extension `.csv` or `.mhp`.
5. Repeat steps 3 and 4 a few times, selecting different files.
6. Repeat steps 3 and 4 a few more times, selecting files you've already added.
7. Click the clear button.

Postconditions

- Upon adding a profile, the label should change from <no profiles loaded> to the full pathname of the selected files, separated by newlines. Adding a duplicate file should fail, with no change in state. Clearing the list should set the label back to <no profiles loaded>.

Results

Works exactly as expected.

2.7 Parameter Selection Panel

Importance

Level 1.

Related Documentation

N/A.

Preconditions

- Test 2.1 passed.
- Test 2.2 passed.
- Test 2.6 passed.
- The user has several profile files.

Procedures

1. Run the command `./build.sh; ./run.sh`.
2. Click the next button to navigate through the panes.
3. On the profile selection pane, click the add profile button.
4. In the dialog box, select a profile file with extension `.csv` or `.mhp`.
5. Repeat steps 3 and 4 a few times, selecting different files.
6. Click the next button to navigate to the parameter selection.
7. Assign some profiles to some parameters.
8. Assign some assigned parameter to the profile "none".
9. Click the clear button.

Postconditions

- Upon adding a parameter assignment, the right pane should reflect the change by updating it's value. Assigning the "none" profile to an assigned parameter should clear it from the right pane. Clicking the clear button should empty the right pane.

Results

Works exactly as expected.

2.8 Final Preview Panel

Importance

Level 3.

Related Documentation

N/A.

Preconditions

- Test 2.1 passed.
- Test 2.2 passed.
- Test 2.6 passed.
- Test 2.7 passed.
- The user has several profile files.

Procedures

1. Run the command `./build.sh; ./run.sh`.
2. Click the next button to navigate through the panes.
3. On the Final Preview Pane, click the Play button to preview the humanized file.
4. Press the "Humanize!" button to go to the next panel to save newly humanized file.
5. Press the "<–Previous" button to return to the previous pane and adjust modifications.

Postconditions

- Upon pressing play, the newly humanized file should play, the button should turn to a stop button, and the progress bar should begin progression. Upon pressing the stop button, the progress bar should reset to 0, the stop button should turn to the play button, and the midi file should stop.

Results

Works exactly as expected.

2.9 Final Save Panel

Importance

Level 1.

Related Documentation

N/A.

Preconditions

- Test 2.1 passed.
- Test 2.6 passed.
- Test 2.7 passed.
- The user has selected at least one parameter to modify.

Procedures

1. Run the command `./build.sh; ./run.sh`.
2. Click the next button to navigate through the panes.
3. On the final save pane, enter a file name and click the save button.
4. In the file chooser select the directory in which to save, and press the save button.
5. Click the previous button to return to the previous panes and manipulate additional files.

Postconditions

- Upon saving the file, the directory in which the file was saved should contain the newly created file.

Results

Works exactly as expected.

3 Input/Output Test Cases

3.1 Logger: Basic string logging

Importance

Level 3.

Related Documentation

N/A.

Preconditions

- Tests in section 1 pass.

Procedures

1. Run the command `./build.sh; ./run.sh`.
2. In a separate terminal window, run the command `tail -f classes/midihumanizer-*.log`.
3. Observe the output.

Postconditions

- Upon saving clicking around in the GUI and interacting with the system, the log file should grow.

Results

Works exactly as expected.

3.2 Logger: Exception logging

Importance

Level 3.

Related Documentation

N/A.

Preconditions

- Tests in section 1 pass.
- Test 3.1 passes.

Procedures

1. Run the command `./build.sh; ./run.sh` from a purposely broken build.
2. In a separate terminal window, run the command `tail -f classes/midihumanizer-*.log`.
3. Observe the output.

Postconditions

- Upon doing something illegal, the Exception thrown should be logged, and then it should pass through to Standard Error.

Results

Works exactly as expected.

3.3 MidiFileHandler: Loading MIDI file

Importance

Level 1.

Related Documentation

N/A.

Preconditions

- Test 2.1 passed.
- Test 2.2 passed.
- The user has a MIDI file.

Procedures

1. Run the command `./build.sh; ./run.sh`.
2. Click the next button to navigate through the panes.
3. On the file selection pane, click the select a file button.
4. In the dialog box, select a MIDI file with extension `.mid` or `.midi`.
5. Click the play button to preview.
6. Click the stop button to stop the preview.

Postconditions

- Upon selecting a file to load, the label should change from <no file selected> to the full pathname of the selected file. Clicking the play button at this point will allow the file to be previewed. While previewing a file, the play button should change to a stop button. Clicking this should stop the preview.

Results

Works exactly as expected.

3.4 ProfileHandler: Profile File Loading

Importance

Level 1.

Related Documentation

N/A.

Preconditions

- Test 2.1 passed.
- Test 2.2 passed.
- The user has several profile files.

Procedures

1. Run the command `./build.sh; ./run.sh`.
2. Click the next button to navigate through the panes.
3. On the profile selection pane, click the add profile button.
4. In the dialog box, select a profile file with extension `.csv` or `.mhp`.
5. Repeat steps 3 and 4 a few times, selecting different files.
6. Repeat steps 3 and 4 a few more times, selecting files you've already added.
7. Click the clear button.

Postconditions

- Upon adding a profile, the label should change from <no profiles loaded> to the full pathname of the selected files, separated by newlines. Adding a duplicate file should fail, with no change in state. Clearing the list should set the label back to <no profiles loaded>.

Results

Works exactly as expected.

4 Statistics Utilities Test Cases

4.1 DiscreteRNG: next()

Importance

Level 1.

Related Documentation

N/A.

Preconditions

- Tests in section 1 pass.

Procedures

1. Run the command `./build.sh; cd classes/; java ladasha.statistics.DiscreteTester`.
2. Interact with the program.

Postconditions

- When generating numbers, they should fall within the discrete distribution provided by the user.

Results

Works exactly as expected.

4.2 GaussianRNG: next()

Importance

Level 1.

Related Documentation

N/A.

Preconditions

- Tests in section 1 pass.

Procedures

1. Run the command `./build.sh; cd classes/; java ladasha.statistics.GaussianTester`.
2. Interact with the program.

Postconditions

- When generating numbers, they should fall within the Gaussian distribution provided by the user.

Results

Works exactly as expected.

5 Humanization Test Cases

5.1 Humanizer: Humanizes

Importance

Level 1.

Related Documentation

N/A.

Preconditions

- Tests in all previous sections pass.

Procedures

1. Run the command `./build.sh; ./run.sh`.
2. In a separate terminal window, run the command `tail -f classes/midihumanizer-*.log`.
3. Observe the output.
4. Load a MIDI file.
5. Assign some profiles.
6. Click next.
7. Click the preview button and listen to the file. It should sound different from the original file.

Postconditions

- Upon humanization, the file should sound different.

Results

Works exactly as expected.

5 Source Code

```
// MIDI Humanizer GUI
// Team La-a
// 20111104

package ladasha.GUI;

import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import javax.swing.*;
import javax.swing.filechooser.*;
import javax.sound.midi.*;
import java.io.*;
import javax.imageio.*;
import java.util.*;
import ladasha.io.*;
import ladasha.statistics.*;
import ladasha.processes.*;

public class GUI {
  // globals, yo
  private static String currentPane = "0"; // first card
  private static int currentPaneIndex = 0; // internal representation
  private static int lastPaneIndex = 6; // last card
  private static JPanel cardpanel; // panel to contain cards
  private static File selectedFile; // the selected file to work on
  private static ArrayList<File> profiles = new ArrayList<File>(); //profiles...
  private static ArrayList<File> stdProfiles = new ArrayList<File>(); //built in profiles
  private static JComboBox<String> profileBox = new JComboBox<String>();
        private static JComboBox<String> trackBox = new JComboBox<String>();
        private static JComboBox<String> parameterBox = new JComboBox<String>();
  private static JButton next; // next button, control panel
  private static JButton prev; // previous button, control panel
  private static JButton help; // help button, control panel
  private static JButton clearModificationsButton = new JButton("Clear");
  private static JLabel bottom; //control panel
  private static JButton assignModificationButton = new JButton("Assign");
  private static JLabel modificationsSelected = new JLabel("<html>\n<center><b> no modificatio
ns selected\n</html>");
  private static BufferedImage image;
  private static BufferedImage image2;
  private static BufferedImage image3;
  private static BufferedImage image4;
  private static BufferedImage image5;
  private static MidiFileHandler midifilehandler = MidiFileHandler.getInstance();
  private static HashMap<String,File> modifications = new HashMap<String,File>();
  private static Humanizer humanizer = Humanizer.getInstance();
        private static boolean hasBeenHumanized = false;
  private static JTextField fileName = new JTextField(25);
        private static JProgressBar humanizingBar = new JProgressBar();
        private static JProgressBar time2 = new JProgressBar();
  private static Logger logger = Logger.getInstance();
  private static Sequencer sequencer;

  private static void createAndShowGUI() {
    /***************************************************************************
     * Frame Initialization code - Don't touch this except for the size
     ***************************************************************************/
    JFrame frame = new JFrame("MIDI Humanizer, by Team La-a");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setIconImage(new ImageIcon("icon.png").getImage());
    frame.setSize(600,450);
    /*************************************************************************
     * All the cards - do your work here
     *************************************************************************
     * Welcome card (index 0)
    *************************************************************************/

    //load images
    try {
      image = ImageIO.read(new File("MIDI.png"));
    } catch (IOException io) {logger.log(io);}

    //instantiate and define settings for components
    JLabel card0label = new JLabel(new ImageIcon(image));
    JPanel card0 = new JPanel(new FlowLayout(FlowLayout.CENTER,200,100));

    //put everything together
    card0.add(card0label);

    /*************************************************************************
     * File selection card (index 1)
     ************************************************************************/

    //load images
    try {
      image4 = ImageIO.read(new File("play.png"));
      image5 = ImageIO.read(new File("stop.png"));
    } catch (IOException io) {logger.log(io);}

                //instantiate and define settings for components
    final JPanel card1 = new JPanel(new BorderLayout());
    final JPanel card1top = new JPanel(new FlowLayout(FlowLayout.CENTER));
    final JPanel card1center = new JPanel(new FlowLayout(FlowLayout.CENTER,500,20));
    final JPanel card1ctop = new JPanel(new FlowLayout(FlowLayout.CENTER));
    final JPanel card1ccenter = new JPanel(new FlowLayout(FlowLayout.CENTER));
    final JPanel card1cbottom = new JPanel(new BorderLayout());
    final JLabel fileLabel = new JLabel("<html><h2>Select a MIDI file to humanize by clicking
the button below</h2></html>");
    final JLabel selectedFileLabel = new JLabel("<html><b>no file selected</html>");
    final JButton fileSelectButton = new JButton("Select a file");
    final JButton initialPreviewButton = new JButton(new ImageIcon(image4));
    final JButton stopButton = new JButton(new ImageIcon(image5));
    final JFileChooser fc = new JFileChooser();
    final JProgressBar time = new JProgressBar();
    time.setPreferredSize(new Dimension(200,22));
    fc.setFileFilter(new FileNameExtensionFilter("MIDI Files","midi", "mid"));
    initialPreviewButton.setEnabled(false);
    stopButton.setVisible(false);

    //assemble components
    card1.add(card1top,BorderLayout.NORTH);
    card1.add(card1center,BorderLayout.CENTER);
    card1top.add(fileLabel);
    card1ctop.add(fileSelectButton);
    card1ccenter.add(selectedFileLabel);
    card1ccenter.add(initialPreviewButton);
    card1ccenter.add(stopButton);
    card1cbottom.add(time,BorderLayout.NORTH);
    card1center.add(card1ctop);
    card1center.add(card1ccenter);
    card1center.add(card1cbottom);

    // add listeners
    fileSelectButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        try{ //show file selection window
          fc.showOpenDialog(card1);
        } catch (HeadlessException ex) {logger.log(ex);}
        selectedFile = fc.getSelectedFile();
        if (selectedFile != null) { //show filename and enable next and previewing
          logger.log("GUI: fileSelectButton: loaded midi file: " + selectedFile.getName());
          selectedFileLabel.setText("<html><b>"
                                + selectedFile.getName()
                                        + "</html>");
                                midifilehandler.loadMidiFile(selectedFile);
                                        initialPreviewButton.setEnabled(true);
                                        time.setStringPainted(true);
          long minutes = midifilehandler.getMidiSequence().getMicrosecondLength()/60000000;
          long seconds = midifilehandler.getMidiSequence().getMicrosecondLength() % 60000000 /
 1000000;
          String s;
          if (seconds < 10) s = "0:00/" + minutes + ":" + 0 + seconds;
          else s = "0:00/" + minutes + ":" + seconds;
          time.setString(s);
                time.setMaximum((int)midifilehandler.getMidiSequence().getMicrosecondLength())
;
                                        clearModificationsButton.doClick();
                                        next.setEnabled(true);
                                }
                        }
                });

    initialPreviewButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        if (!midifilehandler.loadMidiFile(selectedFile)) return; // load file, break on failur
e
        else {
          initialPreviewButton.setVisible(false);
          stopButton.setVisible(true);
                                        sequencer = midifilehandler.getMidiSequencer();
          final String length = midifilehandler.getMidiSequence().getMicrosecondLength()/60000
000 + ":" +
                                                (midifilehandler.getMidiSequence().getMicrosec
ondLength() % 60000000) /1000000;
          midifilehandler.previewUntilStop(); // play preview
          new Thread(new Runnable() {
            public void run() {
                try {
                        while (!midifilehandler.isPlaying()) {
                        Thread.sleep(500);
                  }
                  while (midifilehandler.isPlaying()) {
                    Thread.sleep(100);
                    long minutes = sequencer.getMicrosecondPosition()/60000000;
                    long seconds = sequencer.getMicrosecondPosition() % 60000000 / 1000000;
                    String s;
                    if (seconds < 10) s = minutes + ":" + 0 + seconds + "/" + length;
                    else s = minutes + ":" + seconds + "/" + length;
                    time.setString(s);
                    time.setValue((int)sequencer.getMicrosecondPosition());
                  }

                } catch (InterruptedException ie) {}
                        time.setValue(0);
                                                                time.setString("0:00/" + seque
ncer.getMicrosecondLength()/60000000 +
                                                                        ":" + sequencer.getMic
rosecondLength() % 60000000/ 1000000);
                                                                  stopButton.setVisible(false);
                  initialPreviewButton.setVisible(true);
                }
                }).start();
        }
        }
        });

    stopButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        // stop the loaded file
        new Thread(new Runnable() {
          public void run() {
            midifilehandler.stop();
            initialPreviewButton.setVisible(true);
            stopButton.setVisible(false);
            try {
              while(time.getValue() > 0) {
                time.setValue(0);
                time.setString("0:00/" + midifilehandler.getMidiSequence().getMicrosecondLengt
h()/60000000 +
                                                                          ":" + (midifilehandler
.getMidiSequence().getMicrosecondLength() % 60000000) /1000000);
                Thread.sleep(100);
              }
            } catch (InterruptedException ie) {}
          }
        }).start();
      }
    });



    /****************************************************************************
     * Profile selection card (index 2)
     ***************************************************************************/

    //load images

    //instantiate and define settings for components
    final JPanel card2 = new JPanel(new BorderLayout());
    final JPanel card2top = new JPanel(new FlowLayout(FlowLayout.CENTER));
    final JPanel card2center = new JPanel(new BorderLayout());
    final JPanel card2ctop = new JPanel(new BorderLayout());
                final JPanel card2ctopt = new JPanel(new BorderLayout());
                final JPanel card2ctoptt = new JPanel(new FlowLayout(FlowLayout.CENTER));
                final JPanel card2ctoptb = new JPanel(new FlowLayout(FlowLayout.CENTER));
                final JPanel card2ccenter = new JPanel(new BorderLayout());
    final JPanel card2cbottom = new JPanel(new FlowLayout(FlowLayout.CENTER));
                final JPanel card2bottom = new JPanel(new FlowLayout());
                final JLabel card2label = new JLabel("<html><h2>Add or create your custom prof
iles here</h2></html>");
                final JLabel addLabel = new JLabel("<html><h3>Add a saved profile</h3></html>"
);
    final JLabel loadedProfileList = new JLabel("<html><h3>Custom profiles loaded</h3><p><b>no
 custom profiles loaded</html>", SwingConstants.CENTER);
    final JButton profileAddButton = new JButton("Add profile...");
                final JLabel newProfileName = new JLabel("<html><h3>Enter your profile name</h
3></html>");
                final JLabel newMean = new JLabel("<html><h3>Mean</h3></html>",SwingConstants.
CENTER);
                final JLabel newDeviation = new JLabel("<html><h3>Standard<br>Deviation</h3></
html>",SwingConstants.CENTER);
                final JTextField newProfileNameField = new JTextField(20);
                final JSlider newMeanSlider = new JSlider(-100,100);
                final JSlider newDeviationSlider = new JSlider(0,100,0);
                final JButton createProfileButton = new JButton("Create new");
    final JButton clearProfilesButton = new JButton("Clear");
    final JFileChooser pc = new JFileChooser();
    final JScrollPane card2cbottomScroll = new JScrollPane(card2cbottom,ScrollPaneConstants.VE
RTICAL_SCROLLBAR_AS_NEEDED,ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
                card2cbottomScroll.setPreferredSize(new Dimension(300,85));
                card2cbottomScroll.setBorder(null);
                newMeanSlider.setPreferredSize(new Dimension(500,50));
                newMeanSlider.setMajorTickSpacing(25);
                newMeanSlider.setMinorTickSpacing(5);
                newMeanSlider.setPaintTicks(true);
                newMeanSlider.setPaintLabels(true);
                newDeviationSlider.setPreferredSize(new Dimension(500,50));
                newDeviationSlider.setMajorTickSpacing(25);
                newDeviationSlider.setMinorTickSpacing(5);
                newDeviationSlider.setPaintTicks(true);
                newDeviationSlider.setPaintLabels(true);
    pc.setFileFilter(new FileNameExtensionFilter("MIDI Humanizer Profiles","mhp", "csv"));

    //load built-in profiles
    logger.log("GUI: stdProfiles: loading standard profiles");
    stdProfiles.add(new File("WorldClass.mhp"));
    stdProfiles.add(new File("Pro.mhp"));
    stdProfiles.add(new File("Amateur.mhp"));
    stdProfiles.add(new File("Sophomore.mhp"));
    stdProfiles.add(new File("Newbie.mhp"));
    stdProfiles.add(new File("KidsPlay.mhp"));
    stdProfiles.add(new File("Dis1.mhp"));
    stdProfiles.add(new File("Dis2.mhp"));
    stdProfiles.add(new File("Dis3.mhp"));
    for (File f : stdProfiles)
                        profiles.add(f);

    //assemble components
    card2.add(card2top,BorderLayout.NORTH);
    card2.add(card2center,BorderLayout.CENTER);
                card2.add(card2bottom,BorderLayout.SOUTH);
                card2top.add(card2label);
                card2ctopt.add(card2ctoptt,BorderLayout.NORTH);
                card2ctopt.add(card2ctoptb,BorderLayout.SOUTH);
                card2ctoptb.add(newProfileName);
                card2ctoptb.add(newProfileNameField);
    card2center.add(card2ctop,BorderLayout.NORTH);
                card2center.add(card2ccenter,BorderLayout.CENTER);
    card2center.add(card2cbottomScroll,BorderLayout.SOUTH);
                card2ctoptt.add(addLabel);
    card2ctoptt.add(profileAddButton);
    card2ctoptb.add(createProfileButton);
                card2bottom.add(clearProfilesButton);
    card2cbottom.add(loadedProfileList);
                card2ctop.add(card2ctopt,BorderLayout.NORTH);
                card2ctop.add(newMean,BorderLayout.CENTER);
                card2ctop.add(newMeanSlider,BorderLayout.EAST);
                card2ccenter.add(newDeviation,BorderLayout.CENTER);
                card2ccenter.add(newDeviationSlider,BorderLayout.EAST);

    //add listeners
    profileAddButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
```

```
                                File temp = null;
                                int index = 0;
        try{ //show file selection window
          pc.showOpenDialog(card2);
        } catch (HeadlessException ex) {logger.log(ex);}
        temp = pc.getSelectedFile();
        if (temp != null && !profiles.contains(temp)) {
                        logger.log("GUI: profileAddButton: adding profile " + temp.getName());

                String fileName = temp.getName();
          profiles.add(temp); //add to profiles list
                                profileBox.addItem(fileName); //add to card3's profile selecti
on box
          //show loaded profiles
                                StringBuilder newlabelcontents = new StringBuilder();
          newlabelcontents.append("<html><center><h3>Custom Profiles Loaded</h3>");
          for (File f : profiles) {
                                if (!stdProfiles.contains(f))
              newlabelcontents.append("<p>" + f.getName());
                                }
          newlabelcontents.append("</center></html>");
          loadedProfileList.setText(newlabelcontents.toString());
        }
      }
    });

        createProfileButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                String s = "[" + newMeanSlider.getValue() + "," + newDeviation
Slider.getValue() + "]";
                                logger.log("GUI: createProfileButton: making custom gaussian p
rofile -> " + s);
        File temp = new File(newProfileNameField.getText() + ".mhp");
                                pc.setSelectedFile(temp);
                                int retval = JFileChooser.ERROR_OPTION;
                                try {
                                                retval = pc.showSaveDialog(card2);
                                } catch (HeadlessException ex){logger.log(ex);}
                                if (retval == JFileChooser.APPROVE_OPTION) {
                                        FileWriter outputStream = null;
                                        logger.log("GUI: createProfileButton: saving custom pr
ofile to " + pc.getSelectedFile().getName());
                try{
                                                outputStream = new FileWriter(pc.getSelectedFi
le());
                                                BufferedWriter out = new BufferedWriter(output
Stream);
                                                out.write(s);
                                                out.close();
                                        } catch (IOException ex) { logger.log(ex);}
                                        if (pc.getSelectedFile() != null && !profiles.contains
(pc.getSelectedFile())) {
                                        String fileName = pc.getSelectedFile().getName();
                  profiles.add(pc.getSelectedFile()); //add to profiles list
                                        profileBox.addItem(fileName); //add to card3's profile
 selection box
                  //show loaded profiles
                                        StringBuilder newlabelcontents = new StringBuilder();
                  newlabelcontents.append("<html><center><h3>Custom Profiles Loaded</h3>");
                  for (File f : profiles) {
                                        if (!stdProfiles.contains(f))
                      newlabelcontents.append("<p>" + f.getName());
                                        }
                  newlabelcontents.append("</center></html>");
                  loadedProfileList.setText(newlabelcontents.toString());
                }
                                        JOptionPane.showMessageDialog(createProfileButton,"Pro
file Saved.");
                                }
                        }
                });

                clearProfilesButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        logger.log("GUI: clearProfilesButton: clearing loaded custom profiles");
        for (File f : profiles)
                                if (!stdProfiles.contains(f)) {
                                profileBox.removeItem(f.getName());
                                                if (modifications != null) {
                                                                String key = "";
                                                                while (modifications.containsV
alue(f)) {
                                                                        for (String s : modifi
cations.keySet())
                                                                                if (f == modif
ications.get(s)) {
                                                                                        key = 
s;
                                                                                        break;
                                                                                }
                                                                        modifications.remove(k
ey);
                                                                }
                                                }
                                        }
                                if (!modificationsSelected.getText().contains("no modification
s selected")) {
                                        logger.log("GUI: clearProfilesButton: clearing custom 
profiles from modifications selected");
                                        StringBuilder newlabelcontents = new StringBuilder();
                                        newlabelcontents.append("<html>\n<table cellpadding =\
"2\" cellspacing =\"2\">\n");
                if (modifications.isEmpty()) {
                        newlabelcontents.append("</table><br><center><b>no modifications selec
ted\n");
                  next.setEnabled(false);
                } else {
                  next.setEnabled(true);
                  for (String s : modifications.keySet()) {
                    String key = s;
                    if (key.length() == 2)
                      newlabelcontents.append("<tr><td width = \"30\"><center>" + key.substrin
g(0,1) + "</td>");
                    else newlabelcontents.append("<tr><td width = \"30\"><center>" + key.subst
ring(0,2) + "</td>");
                        if (key.endsWith("V")) newlabelcontents.append("<td width = \"100\"><c
enter>Velocity</td>");
                        if (key.endsWith("T")) newlabelcontents.append("<td width = \"100\"><c
enter>Timing</td>");
                        if (key.endsWith("A")) newlabelcontents.append("<td width = \"100\"><c
enter>Amplitude </td>");
                        newlabelcontents.append("<td width = \"150\"><center>" + modifications
.get(key).getName() + "</td></tr>\n");
                  }
                                                newlabelcontents.append("</html>");
                        modificationsSelected.setText(newlabelcontents.toString());
```

```
                                        }
                                }
                                profiles.clear();
                                next.setEnabled(true);
                                for (File f : stdProfiles)
                                        profiles.add(f);
        loadedProfileList.setText("<html><b>no custom profiles loaded</html>");
                        }
    });



    /************************************************************************
     * Parameter selection card (index 3)
     ************************************************************************/

    //load images


    //instantiate and define settings for components
    final JPanel card3 = new JPanel(new BorderLayout());
    final JPanel card33 = new JPanel(new GridLayout(1,2));
    final JPanel card3top = new JPanel(new FlowLayout(FlowLayout.CENTER));
    final JPanel card3right = new JPanel(new FlowLayout(FlowLayout.CENTER));
    final JPanel card3rightCenter = new JPanel(new FlowLayout(FlowLayout.CENTER));
    final JPanel card3left = new JPanel(new FlowLayout(FlowLayout.CENTER,0,25));
    final JPanel card3leftTop = new JPanel(new FlowLayout(FlowLayout.CENTER,0,50));
    final JPanel card3leftTop1 = new JPanel(new BorderLayout());
    final JPanel card3leftTop2 = new JPanel(new BorderLayout());
    final JPanel card3leftTop3 = new JPanel(new BorderLayout());
    final JPanel card3leftBottom = new JPanel(new FlowLayout(FlowLayout.CENTER,10,0));
    final JLabel card3label = new JLabel("<html><h2>Select Parameters to Modify</h2></html>");
    final JLabel modificationLabel = new JLabel("<html>\n<center><h3>Modifications selected:</
h3>\n<table cellpadding =\"2\" cellspacing =\"2\">\n<tr>" +
                "<td width = \"30\"><center><b>Track</td><td width = \"100\"><center><b>Parame
ter</td><td width = 150><center><b>Profile</td></tr></table></html>");
    final JScrollPane card3rightScroll = new JScrollPane(card3rightCenter,ScrollPaneConstants.
VERTICAL_SCROLLBAR_AS_NEEDED,ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
    card3leftTop.setPreferredSize(new Dimension(275,225));
    card3leftTop1.setPreferredSize(new Dimension(275,22));
    card3leftTop2.setPreferredSize(new Dimension(275,22));
    card3leftTop3.setPreferredSize(new Dimension(275,22));
    card3rightScroll.setBorder(null);
    card3rightScroll.setPreferredSize(new Dimension(300,250));
    assignModificationButton.updateUI();
                clearModificationsButton.updateUI();


    //selectable tracks
    trackBox.setPreferredSize(new Dimension(175,25));
    trackBox.addItem("All");
                trackBox.updateUI();
    for (int i = 1; i <= 16; i++)
      trackBox.addItem("Channel " + i);
    final JLabel trackLabel = new JLabel("<html><b>Channel:  </b></html>",SwingConstants.RIGHT
);

    //selectable parameters
    final JLabel parameterLabel = new JLabel("<html><b>Parameter:  </b></html>",SwingConstants
.RIGHT);
    String[] parameters = new String[4];
    parameters[0] = "All";
    parameters[1] ="Velocity";
    parameters[2] = "Timing";
    parameters[3] = "Amplitude";
            for (int i = 0; i <=3; i++)
        parameterBox.addItem(parameters[i]);
    parameterBox.setPreferredSize(new Dimension(175,25));
    parameterBox.updateUI();

    //selectable profiles, starting with built in mhp’s
    final JLabel profileLabel = new JLabel("<html><b>Profile:  </b></html>",SwingConstants.RIG
HT);
    profileBox.setPreferredSize(new Dimension(175,25));
    profileBox.updateUI();
            profileBox.addItem("none");
    for (File f : stdProfiles)
      profileBox.addItem(f.getName());

    //assemble components
    card3.add(card3top,BorderLayout.NORTH);
    card3.add(card33,BorderLayout.CENTER);
    card33.add(card3left);
    card33.add(card3right);
    card3right.add(modificationLabel);
    card3right.add(card3rightScroll);
    card3rightCenter.add(modificationsSelected);
    card3top.add(card3label);
    card3left.add(card3leftTop);
    card3left.add(card3leftBottom);
    card3leftTop.add(card3leftTop1);
    card3leftTop.add(card3leftTop2);
    card3leftTop.add(card3leftTop3);
    card3leftTop1.add(trackLabel,BorderLayout.CENTER);
    card3leftTop1.add(trackBox,BorderLayout.EAST);
    card3leftTop2.add(parameterLabel,BorderLayout.CENTER);
    card3leftTop2.add(parameterBox,BorderLayout.EAST);
    card3leftTop3.add(profileLabel,BorderLayout.CENTER);
    card3leftTop3.add(profileBox,BorderLayout.EAST);
    card3leftBottom.add(assignModificationButton);
    card3leftBottom.add(clearModificationsButton);

    //add listeners
    assignModificationButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        StringBuilder newlabelcontents = new StringBuilder();
              String selectedProfile = profileBox.getSelectedItem().toString();
              String selectedTrack = trackBox.getSelectedItem().toString();
              String selectedParameter = parameterBox.getSelectedItem().toString();
        File profile = null;
        logger.log("GUI: assignModificationButton: assigning profile " + selectedProfile + " t
o channel " + selectedTrack + " parameter " + selectedParameter);
        for(File f : profiles) {
          if (f.getName().equals(selectedProfile))
            profile = f;
        }
        if (selectedTrack.equals("All")) { //if assigning to all tracks
          if (selectedParameter.toString().equals("All")) { //if assigning to all parameters a
s well
            if (!selectedProfile.equals("none")) { //if not removing assignments
              for (Integer i = 1; i <= 16; i++) { //add assignments to all tracks and paramete
rs
                modifications.put(i.toString() + "V", profile);
                modifications.put(i.toString() + "T", profile);
                modifications.put(i.toString() + "A", profile);
              }
            } else modifications.clear(); //removing all assignments if selected profile == no
ne
          } else { //if assigning to all tracks, but not to all parameters
            for (Integer i = 1; i <= 16; i++) {
              if (selectedProfile.equals("none")) //clear modifications
                modifications.remove(i.toString() + selectedParameter.substring(0,1));
              else //add assignments to all tracks
                modifications.put(i.toString() + selectedParameter.substring(0,1), profile);
            }
          }
        } else if (selectedParameter.equals("All")) { //if assigning to all parameters for a s
ingle track
          if (!selectedProfile.equals("none")) { //if not removing assignments, assign
            modifications.put(selectedTrack.substring(8) + "V", profile);
            modifications.put(selectedTrack.substring(8) + "T", profile);
            modifications.put(selectedTrack.substring(8) + "A", profile);
          } else { //remove assignments
            modifications.remove(selectedTrack.substring(8) + "V");
            modifications.remove(selectedTrack.substring(8) + "T");
            modifications.remove(selectedTrack.substring(8) + "A");
          }
        } else { //if working within a single parameter and single track
          String key = selectedTrack.substring(8) + selectedParameter.substring(0,1);
          if (selectedProfile.equals("none")) modifications.remove(key); //remove if none sele
cted
          else modifications.put(key, profile); //add assignment
        }
        Humanizer.setMidiFile(selectedFile);
        Humanizer.setProfiles(modifications);
        //build table of selected modifications
        newlabelcontents.append("<html>\n<table cellpadding =\"2\" cellspacing =\"2\">\n");
        if (modifications.isEmpty()) {
          newlabelcontents.append("</table><br><center><b>no modifications selected\n");
                next.setEnabled(false);
        } else {
                next.setEnabled(true);
          for (Object o : modifications.keySet()) {
                  String key = o.toString();
                  if (key.length() == 2)
              newlabelcontents.append("<tr><td width = \"30\"><center>" + key.substring(0,1) +
 "</td>");
                  else newlabelcontents.append("<tr><td width = \"30\"><center>" + key.substri
ng(0,2) + "</td>");
                if (key.endsWith("V")) newlabelcontents.append("<td width = \"100\"><center>Ve
locity</td>");
                if (key.endsWith("T")) newlabelcontents.append("<td width = \"100\"><center>Ti
ming</td>");
                if (key.endsWith("A")) newlabelcontents.append("<td width = \"100\"><center>Am
plitude </td>");
                newlabelcontents.append("<td width = \"150\"><center>" + modifications.get(key
).getName() + "</td></tr>\n");
          }
        }
        newlabelcontents.append("</html>");
        modificationsSelected.setText(newlabelcontents.toString());
     }
    });

    clearModificationsButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        logger.log("GUI: clearModificationsButton: clearing assigned modifications");
        modifications.clear();
        modificationsSelected.setText("<html><center><b> no modifications selected\n</html>");
        next.setEnabled(false);
      }
    });

                 /*************************************************************************
     * Interlude card (index 4)
     *************************************************************************/

                final JPanel card4 = new JPanel(new FlowLayout(FlowLayout.CENTER,0,200));
                humanizingBar.setString("Humanizing...");
                humanizingBar.setPreferredSize(new Dimension(400,22));
                humanizingBar.setStringPainted(true);
                humanizingBar.setAlignmentY(SwingConstants.CENTER);
                card4.add(humanizingBar);





     /*************************************************************************
     * Preview card (index 5)
     *************************************************************************/

   //load images

    //instantiate and define settings for components
    final JPanel card5 = new JPanel(new BorderLayout());
    final JPanel card5top = new JPanel(new FlowLayout(FlowLayout.CENTER));
    final JPanel card5center = new JPanel(new FlowLayout(FlowLayout.CENTER,200,50));
    final JLabel card5label = new JLabel("<html><h2>Preview File Before Saving</h2></html>");
    final JButton endPreviewButton = new JButton(new ImageIcon(image4));
    final JButton endStopButton = new JButton(new ImageIcon(image5));
        time2.updateUI();
                time2.setPreferredSize(new Dimension(200,22));
                time2.setString("0:00");

    endStopButton.setVisible(false);

    //assemble components
    card5.add(card5top,BorderLayout.NORTH);
    card5.add(card5center,BorderLayout.CENTER);
    card5top.add(card5label);
    card5center.add(endPreviewButton);
    card5center.add(endStopButton);
                card5center.add(time2);

    //add listeners
    endPreviewButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        logger.log("GUI: endPreviewButton: previewing humanized file");
        if (!midifilehandler.loadMidiFile(selectedFile)) return; // load file, break on failur
e
        else {
                                        final String length = humanizer.getMidiSequence().getM
icrosecondLength()/60000000 +
                                                ":" + humanizer.getMidiSequence().getMicroseco
ndLength() % 60000000 / 1000000;
          endPreviewButton.setVisible(false);
          endStopButton.setVisible(true);
          humanizer.previewStart(); // play preview
          new Thread(new Runnable() {
            public void run() {
```

```
                                                            sequencer = humanizer.getMidiSequencer
();
              try {
                while (!humanizer.isPlaying()) {
                  Thread.sleep(500);
                }
                while (humanizer.isPlaying()) {
                  Thread.sleep(100);
                                                                            long minutes = sequenc
er.getMicrosecondPosition()/60000000;
                    long seconds = sequencer.getMicrosecondPosition() % 60000000 / 1000000;
                    String s;
                    if (seconds < 10) s = minutes + ":" + 0 + seconds + "/" + length;
                    else s = minutes + ":" + seconds + "/" + length;
                    time2.setString(s);
                    time2.setValue((int)humanizer.getMidiSequencer().getMicrosecondPosition())
;
                }
                        } catch (InterruptedException ie) {logger.log(ie);}
              endStopButton.setVisible(false);
              endPreviewButton.setVisible(true);
                                                            time2.setValue(0);
                                                            time2.setString("0:00/" + sequencer.ge
tMicrosecondLength()/60000000 +
                                                                    ":" + sequencer.getMicrosecond
Length() % 60000000/ 1000000);

            }
          }).start();
        }
      }
    });

    endStopButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        logger.log("GUI: endStopButton: stopping humanized file preview");
        // stop the loaded file
        humanizer.previewStop();
                                    new Thread(new Runnable() {
                                            public void run() {
                                                    while (time.getValue() > 0) {
                                                            time2.setValue(0);
                                                            time2.setString("0:00/" + sequencer.ge
tMicrosecondLength()/60000000 +
                                                                    ":" + sequencer.getMicrosecond
Length() % 60000000/ 1000000);
                                                            try {
                                                                    Thread.sleep(100);
                                                            } catch (InterruptedException ie) {log
ger.log(ie);}
                                                    }
                                            }
                                    }).start();
        endStopButton.setVisible(false);
        endPreviewButton.setVisible(true);
      }
    });

    /*****************************************************************************
     * Finalize/save card (index 5)
     ****************************************************************************/

    //load images
    //instantiate and define settings for components
    final JPanel card6 = new JPanel();
    final JLabel card6label = new JLabel("<html><h1>Save the Modified MIDI Sequence<h1><html>"
);
    final JButton saveButton = new JButton("Save As...");

    //assemble components
    card6.add(card6label);
    card6.add(fileName);
    card6.add(saveButton);

    //add listeners
    saveButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        logger.log("GUI: saveButton: saving humanized file");
        File newFile = new File(selectedFile,fileName.getText()); //temp patch, must perform o
perations on file
        fc.setSelectedFile(newFile);
        int retval = JFileChooser.ERROR_OPTION;
        try{ //show file selection window
          //fc.showSaveDialog(card5);
          retval = fc.showSaveDialog(card6);
        } catch (Exception ex) {logger.log(ex);}
        //if (fc.getSelectedFile() != null) // will never catch
        if(retval == JFileChooser.APPROVE_OPTION) { // only save if approved
          humanizer.saveFile(fc.getSelectedFile().toString());
                                        JOptionPane.showMessageDialog(createProfileButton,"New
 MIDI Saved.");
                                }
      }
    });

     /***************************************************************************
     * cards (help pages) index = prevpage + 10
     ***************************************************************************
    * Welcome help card (index 10)
     **************************************************************************/

    //load images

    //instantiate and define settings for components
    JPanel card10 = new JPanel();
    JLabel card10label = new JLabel("<html><h1>About MIDI Humanizer<h1>" +
    "<p>Welcome to MIDI Humanizer, brought to you by Team La-a!" +
    "<p>MIDI Humanizer is a production by:" +
    "<ul><li>Jamie White</li><li>Jason Shaffner</li><li>Jacob Peck</li></ul>" +
    "</html>");

    //assemble components
    card10.add(card10label);

    //add listeners

    /***************************************************************************
     * File Selection help card (index 11)
     **************************************************************************/

    //load images

    //instantiate and define settings for components
    JPanel card11 = new JPanel();
    JLabel card11label = new JLabel("<html><h1>File Selection<h1>" +
    "<p>To get started, load in a MIDI file to work on.<p>MIDI files typically" +
    "have a file extension of .mid or .midi.<p>After you’ve selected a MIDI " +
    "sequence to work<br>on, you may preview it with the play button, and stop the " +
    "preview with the stop button.<p>When you are satisfied, you may continue by " +
    "clicking the next--> button.</html>");

    //assemble components
    card11.add(card11label);

    //add listeners

    /***************************************************************************
     * Profile Selection card (index 12)
     ***************************************************************************/

    //load images

    //instantiate and define settings for components
    JPanel card12 = new JPanel();
    JLabel card12label = new JLabel("<html><h1>Profile Selection<h1>" +
    "<p>On this card, you may load in multiple profiles from which<br>to " +
    "apply variance to various parameters of the MIDI Sequence<br>loaded in " +
    "the previous step.<p>" +
    "To add a profile, click the add button.  Profile files typically<br>have " +
    "a file extension of .mhp or .csv.<p>" +
    "When you have loaded all the profiles you wish to use, click next--></html>");

   //assemble components
    card12.add(card12label);

    //add listeners

    /***************************************************************************
     * Parameter Selection help card (index 13)
     ***************************************************************************/

    //load images

    //instantiate and define settings for components
    JPanel card13 = new JPanel();
    JLabel card13label = new JLabel("<html><h1>Modifications Selection<h1>" +
    "<p>On this card, you may assign the custom profiles you loaded in <br>" +
    "or the built in profiles to various parameters of the MIDI Sequence<br><p>" +
    "To assign a parameter modification, click the assign button.<br>If the profile " +
    "selected is \"None,\" the modifications to <br>whichever parameters are " +
    "selected will be removed.<br>Press the clear button to unassign all modifications.<p>" +
    "When you have assigned all the modifications you wish to use, click next--></html>");

    //assemble components
    card13.add(card13label);

    //add listeners

    /***************************************************************************
     * Preview help card (index 14)
     ***************************************************************************/

    //load images

    //instantiate and define settings for components
    JPanel card15 = new JPanel();
    JLabel card15label = new JLabel("<html><h1>Preview Before Saving<h1>" +
    "<p>Really? You need help for this?<br>Press the play button to preview " +
    "the MIDI Sequence<br>with the modifications you selected in the previous " +
    "step.<br>Press the stop button to stop playing the sequence.<br>" +
    "Go back to change which modifications to apply<br>by pressing the " +
    "previous button,<br>or continue by pressing \"Humanize!\"</html>");

    //assemble components
    card15.add(card15label);

    //add listeners

    /****************************************************************************
     * Finalize/Save help card (index 15)
     ***************************************************************************/

    //load images

    //instantiate and define settings for components
    JPanel card16 = new JPanel();
    JLabel card16label = new JLabel("<html><h1>Save the Modified MIDI Sequence<h1><p>" +
    "Press the save button to choose a directory and<br>filename to save the modified " +
    "MIDI sequence");

    //assemble components
    card16.add(card16label);

    //add listeners


    // assemble all the cards into the panel
    cardpanel = new JPanel(new CardLayout());
    cardpanel.add(card0,"0");
    cardpanel.add(card1,"1");
    cardpanel.add(card2,"2");
    cardpanel.add(card3,"3");
    cardpanel.add(card4,"4");
    cardpanel.add(card5,"5");
                cardpanel.add(card6,"6");
    cardpanel.add(card10,"10");
    cardpanel.add(card11,"11");
    cardpanel.add(card12,"12");
    cardpanel.add(card13,"13");
    cardpanel.add(card15,"15");
    cardpanel.add(card16,"16");


    /****************************************************************************
     * Control Panel Code (bottom button panel)
     ***************************************************************************/

    //load images
    try {
      image2 = ImageIO.read(new File("MIDI2.png"));
      image3 = ImageIO.read(new File("help.png"));
    } catch (IOException io) {logger.log(io);}

    //instantiate and define settings for components
    JPanel controlPanel = new JPanel(new BorderLayout());
    JPanel controls = new JPanel(new FlowLayout(FlowLayout.CENTER));
    next = new JButton("Next-->");
    prev = new JButton("<--Previous");
    bottom = new JLabel(new ImageIcon(image2));
    help = new JButton(new ImageIcon(image3));
    //assemble components
    controlPanel.add(help,BorderLayout.WEST);
    controls.add(prev);
    controls.add(next);
    controlPanel.add(controls,BorderLayout.CENTER);
    controlPanel.add(bottom,BorderLayout.EAST);

    // add listeners for controls
    next.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        CardLayout cl = (CardLayout)(cardpanel.getLayout());
        currentPaneIndex++;
        if (currentPaneIndex > lastPaneIndex) currentPaneIndex = lastPaneIndex;
        currentPane = Integer.toString(currentPaneIndex);
        cl.show(cardpanel, currentPane);
        initializeCard(currentPaneIndex);
      }
    });

    prev.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        CardLayout cl = (CardLayout)(cardpanel.getLayout());
        if (currentPaneIndex <= lastPaneIndex) {
          currentPaneIndex--;
                                  if (currentPaneIndex == 4)
                                          currentPaneIndex = 3;
                          } else
          currentPaneIndex -= 10;
        if (currentPaneIndex < 0) currentPaneIndex = 0;
        currentPane = Integer.toString(currentPaneIndex);
        cl.show(cardpanel, currentPane);
        initializeCard(currentPaneIndex);
      }
    });

    help.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        CardLayout cl = (CardLayout)(cardpanel.getLayout());
        currentPaneIndex += 10;
        if (currentPaneIndex >= lastPaneIndex+10) currentPaneIndex = lastPaneIndex+10;
        currentPane = Integer.toString(currentPaneIndex);
        cl.show(cardpanel, currentPane);
        initializeCard(currentPaneIndex);
      }
    });

    /****************************************************************************
     * GUI Initialization Code
     * Perform any tasks that need to be done before the GUI is shown here
     ***************************************************************************/

    // build the gui, finally
    frame.getContentPane().add(cardpanel, BorderLayout.CENTER);
    frame.getContentPane().add(controlPanel, BorderLayout.SOUTH);

    // start on correct card
    CardLayout cl = (CardLayout)(cardpanel.getLayout());
    cl.show(cardpanel,currentPane);
    initializeCard(currentPaneIndex);

    // display the frame
    //frame.pack();
    frame.setVisible(true);
    frame.setResizable(false);
    frame.setLocationRelativeTo(null); // center on screen
  }

  /*******************************************************************************
   * initializeCard() Method
   * this is called whenever a new card is placed on the screen
   * be careful with this, and don’t touch the breaks
   ******************************************************************************/

  static public void initializeCard(int index) {
    logger.log("GUI: initializeCard(index): initializing card " + index);
    switch(index) {
      case 0:
        next.setText("Let’s Get Started");
        prev.setText("<--Previous");
        next.setEnabled(true);
        next.setVisible(true);
        prev.setEnabled(false);
        prev.setVisible(false);
        help.setEnabled(true);
        help.setVisible(true);
        bottom.setVisible(false);
        midifilehandler.stop();
        break;
      case 1:
        next.setText("Next-->");
        prev.setText("<--Previous");
        if (selectedFile == null)
          next.setEnabled(false);
        next.setVisible(true);
        prev.setEnabled(true);
        prev.setVisible(true);
        help.setEnabled(true);
        help.setVisible(true);
        bottom.setVisible(true);
        midifilehandler.stop();
        break;
      case 2 :
        next.setText("Next-->");
        prev.setText("<--Previous");
        next.setEnabled(true);
        next.setVisible(true);
        prev.setEnabled(true);
        prev.setVisible(true);
        help.setEnabled(true);
        help.setVisible(true);
        bottom.setVisible(true);
        midifilehandler.stop();
        break;
     case 3 :
        next.setText("Next-->");
        prev.setText("<--Previous");
        if (modifications.isEmpty())
                next.setEnabled(false);
        next.setVisible(true);
        prev.setEnabled(true);
        prev.setVisible(true);
        help.setEnabled(true);
        help.setVisible(true);
        bottom.setVisible(true);
        hasBeenHumanized = false;
                                midifilehandler.stop();
                humanizer.previewStop();
        break;
                 case 4:
                                next.setVisible(false);
                                prev.setVisible(false);
                                help.setVisible(false);
                                humanizer.go();
                                while (humanizer.getPasscount() == 0) {
                                        try {
                                                Thread.sleep(50);
                                        } catch (InterruptedException ie) {}
                                }
                humanizingBar.updateUI();
                                humanizingBar.setMaximum(humanizer.getPasscount());
                                humanizingBar.setValue(0);
                                new Thread (new Runnable() {
                                        public void run() {

                                        try {
                                                Thread.sleep(100);
                                                while (humanizingBar.getValue() < humanizingBa
r.getMaximum()) {
                                                        humanizingBar.setValue(humanizer.getPa
ssesCompleted());
              humanizingBar.setString("Humanizing... pass " + humanizingBar.getValue()
                                     + " of " + humanizingBar.getMaximum());
                                                        Thread.sleep(100);
                                                }
                                        } catch (InterruptedException ie) {}
                                        next.doClick();
                                        }
                                }).start();
                                break;
     case 5:
        next.setText("Humanize!");
        prev.setText("<--Previous");
        next.setEnabled(true);
        next.setVisible(true);
        prev.setEnabled(true);
        prev.setVisible(true);
        help.setEnabled(true);
        help.setVisible(true);
        bottom.setVisible(true);
        midifilehandler.stop();
                                new Thread(new Runnable() {
          public void run() {
            //humanizer.go();
                                                long minutes = humanizer.getMidiSequence().get
MicrosecondLength()/60000000;
            long seconds = humanizer.getMidiSequence().getMicrosecondLength() % 60000000 / 100
0000;
            String s;
            if (seconds < 10) s = "0:00/" + minutes + ":" + 0 + seconds;
            else s = "0:00/" + minutes + ":" + seconds;
            time2.setString(s);
                                                time2.setMaximum((int)humanizer.getMidiSequenc
e().getMicrosecondLength());
                                                time2.setStringPainted(true);
          }
        }).start();
                                humanizer.previewStop();
                                break;
      case 6:
        prev.setText("<--Previous");
        next.setEnabled(false);
        next.setVisible(false);
        prev.setEnabled(true);
        prev.setVisible(true);
        help.setEnabled(true);
        help.setVisible(true);
        bottom.setVisible(true);
        hasBeenHumanized = true;
                                fileName.setText(selectedFile.getName().substring(0,selectedFi
le.getName().lastIndexOf(".")) + "-new.mid");
        midifilehandler.stop();
        humanizer.previewStop();
        break;
      case 10: case 11: case 12: case 13: case 14: case 15:
        prev.setText("Close Help");
        next.setEnabled(false);
        next.setVisible(false);
        prev.setEnabled(true);
        prev.setVisible(true);
        help.setEnabled(false);
        help.setVisible(false);
        bottom.setVisible(true);
        break;
      default:
        // shouldn’t get here
        System.out.println("How did you end up here?  Index: " + index);
        bottom.setVisible(true);
        break;
    }
  }

  /*****************************************************************************
   * main() Method - Don’t touch this
   ****************************************************************************/
  static public void main(String[] args) {
    logger.log("GUI: main(args): setting look and feel");
    // Use the system’s Look and Feel
    try {
      UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    } catch (Exception ex) {
      logger.log(ex);
    }

    logger.log("GUI: main(args): creating and showing GUI");
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        createAndShowGUI();
      }
    });
  }
}
```

```
// Logger class

package ladasha.io;

import java.io.*;
import java.util.*;
import java.text.*;

public class Logger {
  // singleton
  private static Logger instance = null;

  private Logger() {
    initialize();
  }

  public static Logger getInstance() {
    return getInstance(DEBUG);
  }

  public static Logger getInstance(int level) {
    if (instance == null) {
      instance = new Logger();
      instance.verbosity = level;
    }
    return instance;
  }

  // fields
  private File logfile;
  private FileWriter writer;
  private Date date = new Date();
  private SimpleDateFormat timestampformat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
  private static int verbosity;

  public static final int CRITICAL = 1;
  public static final int ERROR = 2;
  public static final int DEBUG = 3;
  public static final int NORMAL = 4;
  public static final int VERBOSE = 5;


  // methods
  private void initialize() {
    SimpleDateFormat dateformatYYYYMMDD = new SimpleDateFormat("yyyyMMdd");
    logfile = new File("midihumanizer-" + dateformatYYYYMMDD.format(date).toString() + ".log")
;
    try {
      boolean append = logfile.createNewFile(); // create new file if necessary
      writer = new FileWriter(logfile, true); // append if already exists
      writer.write("MIDI Humanizer - Logfile\n");
      log("Logging started... appending? " + append);
    } catch (Exception ex) {ex.printStackTrace();}
  }

  private String getTimestamp() {
    return timestampformat.format(Calendar.getInstance().getTime()).toString() + " -> ";
  }

  private String getStackTraceAsString(Throwable contents) {
    Writer result = new StringWriter();
    PrintWriter printWriter = new PrintWriter(result);
    contents.printStackTrace(printWriter);
    return result.toString();
  }

  private String getLevel(int level) {
    switch (level) {
      case CRITICAL:
        return "[CRITICAL] ";
      case ERROR:
        return "[ERROR] ";
      case DEBUG:
        return "[DEBUG] ";
      case NORMAL:
        return "[NORMAL] ";
      case VERBOSE:
        return "[VERBOSE] ";
      default:
        return "[UNKNOWN LEVEL: " + level + "] ";
    }
  }

  public void log(String contents) {log(DEBUG, contents);}
  public void log(int level, String contents) {
    if (level > verbosity) return;
    try {
      writer.write(getLevel(level) + getTimestamp() + contents + "\n");
      writer.flush(); // push it to disk
    } catch (Exception ex) {ex.printStackTrace();}
  }

  public void log(Throwable contents) {log(CRITICAL, contents);}
  public void log(int level, Throwable contents) {
    if (level > verbosity) return;
    try {
      writer.write(getLevel(level) + getTimestamp() + "Exception thrown! Details follow:\n");
      writer.write(getStackTraceAsString(contents));
      writer.flush();
      contents.printStackTrace(); // dump to standard error
    } catch (Exception ex) {ex.printStackTrace();}
  }
}

/*
class LoggerTester {
  public static void main(String[] args) throws Exception {
    Logger logger = Logger.getInstance();
    logger.log("test");
    Thread.sleep(2000);
    logger.log("test2");
    logger.log(new Exception("This is a test exception."));
    System.exit(-1);
  }
}
*/
```

```
package ladasha.io;

import java.io.File;
import javax.sound.midi.MidiSystem;
import javax.sound.midi.Sequence;
import javax.sound.midi.Sequencer;


public class MidiFileHandler {
  private File file = null;
  private Sequencer seq1 = null;
  private Sequence seq = null;
  private static MidiFileHandler instance = null;
  private static Logger logger = Logger.getInstance();

  private MidiFileHandler() {
    new Thread(new Runnable() {
      public void run() {
        try {
          seq1 = MidiSystem.getSequencer();
          if(seq1 == null) return; // throw exception here in the future
          else seq1.open();
        } catch (Exception ex) { logger.log(ex);}
      }
    }).start();
  }

  public static MidiFileHandler getInstance() {
    logger.log("obtaining instance of MidiFileHandler");
    if(instance == null) instance = new MidiFileHandler();
    return instance;
  }

  public boolean loadMidiFile(File cake) {
    logger.log("MidiFileHandler: loadMidiFile(cake): loading MIDI file");
    this.file = cake;
    if(file == null) return false; // throw exception here in the future
    try{
      // sequencer gets file
      seq =  MidiSystem.getSequence(cake);
      seq1.setSequence(seq);
    } catch(Exception ex) {
      logger.log(ex);
    }
    return true;
  } // end method

  public Sequence getMidiSequence() {
    logger.log("MidiFileHandler: getMidiSequence(): getting MIDI sequence");
    if(seq == null)
      return null; // throw exception here in the future
    else
      return seq;
  }

        public Sequencer getMidiSequencer() {
                logger.log("MidiFileHandler: getMidiSequencer(): getting MIDI sequencer");
                if(seq1 == null) return null;
                else return seq1;
        }

  public int getTracks() {
    logger.log("MidiFileHandler: getTracks(): getting number of tracks");
    return seq.getTracks().length;
  }

  public void stop() {
    logger.log("MidiFileHandler: stop(): stopping MIDI sequence");
    if(seq1 == null) return; // throw exception here in the future
    try {
      seq1.stop();
    } catch (Exception ex) { logger.log(ex); }
  }

  public boolean isPlaying() {
    return seq1.isRunning();
  }

  public void preview(int millis) {
    logger.log("MidiFileHandler: preview(millis): previewing for " + millis + " milliseconds")
;
    if(seq1 == null) return; // throw exception here in the future
    try{
      Thread t = new Thread(new Runnable() {
        public void run() {
          seq1.start();
        }
      });
      t.start();
      // lets the song get previewed for (millis/1000) sec
      Thread.sleep(millis);
      seq1.stop();
    } catch(Exception ex){
      logger.log(ex);
    }
  }

  public void preview() {
    preview(10000);
  }

  public void previewUntilStop() {
    logger.log("MidiFileHandler: previewUntilStop(): previewing MIDI sequence");
    if(seq1 == null) return; // throw exception here in the future
    try{
      Thread t = new Thread(new Runnable() {
        public void run() {
          seq1.start();
        }
      });
      t.start();
      } catch(Exception ex){
      ex.printStackTrace();
    }
  }
}// end class
```

```
package ladasha.io;

import java.util.*;
import java.io.*;

public class ProfileHandler {

        private static File profile;
        private static HashMap<Double,Double> map = new HashMap<Double,Double>();
        private static ProfileHandler instance = null;
  private static Scanner sc;
  private static Logger logger = Logger.getInstance();

        private ProfileHandler() {}

        public static ProfileHandler getInstance() {
    logger.log("obtaining instance of ProfileHandler");
                if (instance == null) instance = new ProfileHandler();
                return instance;
        }

        public static void setProfile(File newProfile) {
                logger.log("ProfileHandler: setProfile(newProfile): setting profile");
    if (newProfile == null) return;
                profile = newProfile;
        }

        public static void reader() throws IOException {
                logger.log("ProfileHandler: reader(): reading profile");
    int count = 0;
                StringBuilder builder = new StringBuilder(count);
                try {
                  sc = new Scanner(new FileInputStream(profile));
                } catch (FileNotFoundException f) {logger.log(f);}
                sc.useDelimiter(",");
                try {
                        while (sc.hasNext()) {
                                count++;
                                builder.append(sc.nextDouble() + "\n");
                        }
                } catch (Exception exception) { logger.log(exception);
                } finally { sc.close(); }
                System.out.print(builder);
                System.out.print(count);
        }

        public static boolean isGaussian() {
          logger.log("ProfileHandler: isGaussian(): checking if gaussian");
    try {
                        sc = new Scanner(new FileInputStream(profile));
                } catch (FileNotFoundException f) { logger.log(f);}
                String s = sc.nextLine();
                sc.close();
    return s.contains("[");
        }

        public static boolean isDiscrete() {
                logger.log("ProfileHandler: isDiscrete(): checking if discrete");
    try {
                        sc = new Scanner(new FileInputStream(profile));
                } catch (FileNotFoundException f) {logger.log(f);}
                String s = sc.nextLine();
    sc.close();
```

```
                return !s.contains("[");
        }

        public static HashMap<Double,Double> getDiscrete() {
                logger.log("ProfileHandler: getDiscrete(): getting discrete distribution");
    if (!isDiscrete()) return null;
                // variables
                int count = 0;
                double key = 0;
                double value = 0;
                try {
                  sc = new Scanner(new FileInputStream(profile));
                } catch (FileNotFoundException f) { logger.log(f); }
                sc.useDelimiter(",");
                // if count is divisible by 2
                while (sc.hasNext()) {
                        double token = sc.nextDouble();
                        if (count % 2 != 0) { // value, not a key
                                map.put(key, token);
                        } else key = token;
                        count++;
                }
                for (Double d : map.keySet()) {
                  value += map.get(d);
            map.put(d,value);
                }
    sc.close();
    if (0.999999 > value || value > 1.000001) return null; // ugly, incorrect, but better.
                else return map;
        }

        public static Double[] getGaussian() {
                logger.log("ProfileHandler: getGaussian(): getting gaussian distribution");
        if (!isGaussian()) return null;
                try {
                  sc = new Scanner(new FileInputStream(profile));
                } catch (FileNotFoundException f) { logger.log(f);}
    sc.useDelimiter(",");
                String mu = sc.next();
    String sigma = sc.next();
    if (mu == null || sigma == null) return null;
                Double[] meanAndVariance = new Double[2];
                meanAndVariance[0] = Double.parseDouble(mu.substring(mu.lastIndexOf("[")+1));
                meanAndVariance[1] = Double.parseDouble(sigma.substring(0,sigma.length()-1));
                sc.close();
    return meanAndVariance;
        }
}

/*
class ProfileHandlerTester {

        public static void main (String[] args) {
                ProfileHandler p = ProfileHandler.getInstance();
                ProfileHandler.setProfile(new File("˜/humanizer-current/classes/StdGaus1.mhp")
);
                ProfileHandler.getGaussian();
        }
}
*/
```

```java
package ladasha.statistics;

import java.util.*;
import java.io.*;
import ladasha.io.*;

public class DiscreteRNG {

  private static DiscreteRNG instance = null;
  /*
  private static ArrayList<Double> probability;
  private static ArrayList<Double> distribution;
  */
  private static HashMap<Double,Double> probabilities;
  private static Logger logger = Logger.getInstance();

  public static DiscreteRNG getInstance() {
    logger.log("obtaining instance of DiscreteRNG");
    if (instance == null)
      instance = new DiscreteRNG();
    return instance;
  }

  private DiscreteRNG(){}

  /*
  public static void setProbability(ArrayList<Double> newProbability) {
    probability = newProbability;
  }

  public static void setDistribution(ArrayList<Double> newDistribution) {

  }
  */

  public static void setProbabilities(HashMap<Double,Double> newProbabilities) {
    logger.log("DiscreteRNG: setProbabilities(newProbabilities): setting new probabilites");
    probabilities = newProbabilities;
  }

  /*
  public double next() {
    Random r = new Random();
    double temp = r.nextDouble();
    for(Double d : probability) {
      if (temp <= d/1.0) {
        temp = distribution.get(probability.indexOf(d));
              break;
      }
    }
    return temp;
  }
  */

  public static double next() {
    Random r = new Random();
    double temp = r.nextDouble();
    for(Double d : probabilities.keySet()) {
      if (temp <= probabilities.get(d)/1.0) {
        temp = d;
              break;
      }
    }
    logger.log("DiscreteRNG: next(): returning " + temp);
    return temp;
  }
}

/*
class DiscreteTester {

        static DiscreteRNG d = DiscreteRNG.getInstance();
        static Scanner sc = new Scanner(System.in);
        static HashMap<Double,Double> probably = new HashMap<Double,Double>();

        static void prompt() {
                System.out.print("File name >");
                String s = sc.next();
                try {
                  sc = new Scanner(new FileInputStream(new File(s)));
                } catch (FileNotFoundException f) {}
                double running = 0.0;
                sc.useDelimiter(",");
                while (sc.hasNext()) {
                  double key = sc.nextDouble();
                  double value = sc.nextDouble();
                  running += value;
                  probably.put(key,value);
                }
                running = 0.0;
                for (Double d : probably.keySet()) {
                  running += probably.get(d);
                  probably.put(d,running);
                }
                DiscreteRNG.setProbabilities(probably);
                System.out.println("Total probability: " + running);
        }

        static void shoot() {
                for (int i = 0; i < 20; i++)
                        System.out.println(DiscreteRNG.next());
        }


        public static void main(String[] args) {
                prompt();
                shoot();
        }
}
*/
```

```
package ladasha.statistics;

import java.util.*;
import ladasha.io.*;

public class GaussianRNG {

  private static boolean previous;
  private static boolean previous2;
  private static double c;
  private static double t1;
  private static double t2;
  private static double g;
  private static double c1;
  private static Random r;
  private static GaussianRNG instance;
  private static Logger logger = Logger.getInstance();

  public static GaussianRNG getInstance() {
    logger.log("obtaining instance of GaussianRNG");
    if (instance == null) instance = new GaussianRNG();
    return instance;
  }

  private GaussianRNG() {
    previous = false;
    previous2 = false;
    r = new Random();
  }

  public static double next() {
    if (previous) {
      previous = false;
      logger.log("GaussianRNG: next(): returning " + c);
      return c;
    } else {
      double s;
      do {
        t1 = r.nextDouble() * 2 - 1;
        t2 = r.nextDouble() * 2 - 1;
        s = (t1 * t1) + (t2 * t2);
      } while (s >= 1 || s == 0);
      g = Math.sqrt(-2 * Math.log(s)/s);
      c = t1 * g;
      c1 = t2 * g;
      previous = true;
      previous2 = true;
      logger.log("GaussianRNG: next(): returning " + c1);
      return c1;
    }
  }

  public static double next(double mean, double variance) {
    double retval = next() * variance + mean;
    logger.log("GaussianRNG: next(mean,variance): returning " + retval);
    return retval;
  }
}
/*
  public double nextCorrecting(double mean, double variance) {
    if (previous2) {
      previous = false;
      double variance2 = (c1 - mean == 0) ? 1 : (c1);
      return next(((c1 + mean)/2), variance/variance2);
    } else return next();
  }
}


class GaussianTester {

        static GaussianRNG g;
        static Scanner sc = new Scanner(System.in);
        static Random r = new Random();
        static double mean;
        static double variance;

        public static void seuss() {
                System.out.println("My code:");
                g.getInstance();
                for (int i = 0; i < 20; i++) {
                        int j = (int)(10 * g.next());
                        System.out.println(j);
                }
        }

        public static void gauss() {
                System.out.println("Java Random:");
                g.getInstance();
                for (int i = 0; i < 20; i++) {
                        int j = (int)(10 * r.nextGaussian());
                        System.out.println(j);
                }
        }

        public static void greenEggsAndHam() {
                g.getInstance();
                System.out.print("Enter mean >");
                mean = sc.nextInt();
                System.out.print("Enter variance >");
                variance = sc.nextInt();
                System.out.println("Gaussian...");
                for (int i = 0; i < 20; i++) {
                        int j = (int)(g.next(mean,variance));
                        System.out.println(j);
                }
                System.out.print("Enter mean >");
                mean = sc.nextInt();
                System.out.print("Enter variance >");
                variance = sc.nextInt();
                System.out.println("Correcting...");
                for (int i = 0; i < 20; i++) {
                        int j = (int)(g.nextCorrecting(mean,variance));
                        System.out.println(j);
                }
        }

        public static void main(String[] args) {
                seuss();
                gauss();
                while (true) {
                        greenEggsAndHam();
                        System.out.print("Do you wish to exit? y or n? >");
                        if (sc.next().equalsIgnoreCase("y"))
                                break;
                }
        }
}
*/
```

```java
package ladasha.processes;

import java.io.*;
import javax.sound.midi.*;
import java.util.*;
import ladasha.statistics.*;
import ladasha.io.*;

public class Humanizer {
  // singleton
  private static Humanizer instance = null;

  private Humanizer() {}

  public static Humanizer getInstance() {
    logger.log("obtaining instance of Humanizer");
    if (instance == null) instance = new Humanizer();
    return instance;
  }

  public static void setMidiFile(File newFile) {
    logger.log("Humanizer: setMidiFile(newFile): setting MIDI file");
    if (newFile == null) return;
    midiFile = newFile;
  }

  public static void setProfiles(HashMap<String,File> newProfiles) {
    logger.log("Humanizer: setProfiles(newProfiles): setting profile list");
    if (newProfiles == null) return;
    profiles = newProfiles;
    // debug
    /*
    for(String s : profiles.keySet()) {
      System.out.println(s + " -> " + profiles.get(s).toString());
    }
    */
  }

  // fields
  private static File midiFile = null;
  private static HashMap<String,File> profiles = null;
  private Sequence sequenceOrig = null;
  private Sequence sequenceNew = null;
  private Sequencer seq1 = null;
  private static DiscreteRNG discreteRNG = DiscreteRNG.getInstance();
  private static GaussianRNG gaussianRNG = GaussianRNG.getInstance();
  private static ProfileHandler profileHandler = ProfileHandler.getInstance();
  private static Double[] gaussian = new Double[2];
  private static Logger logger = Logger.getInstance();

  private static final int VELO_MASK = 0b1001;
  private static final int AMP_MASK = 0b1010;
  private static final int TIME_MASK = VELO_MASK; // same events

  private static volatile int passcount = 0;
  private static volatile int passescompleted = 0;

  private static volatile boolean initialized = false;

  public static int getPasscount() {
    return passcount;
  }
  public static int getPassesCompleted() {
    return passescompleted;
  }

  public static HashMap<String,File> getProfiles() {
    return profiles;
  }

  public static File getMidiFile() {
    return midiFile;
  }

        public Sequence getMidiSequence() {
                if (sequenceNew == null) return null;
                else return sequenceNew;
        }

        public Sequencer getMidiSequencer() {
                if (seq1 == null) return null;
                else return seq1;
        }
  // methods
  public int go() {
    passcount = 0;
                new Thread(new Runnable() {
      public void run() {
        logger.log("Humanizer: go(): going");
        modifySequence();
      }
    }).start();
    for(;;) {
      //System.out.println("initialized: " + initialized);
      if(initialized) break;
    }
    return getPasscount();
  }

  public void previewStop() {
    logger.log("Humanizer: previewStop(): stopping preview");
    if(seq1 == null) return; // throw exception here in the future
    try {
      seq1.stop();
    } catch (Exception ex) { ex.printStackTrace(); }
  }

  public void previewStart() {
    logger.log("Humanizer: previewStart(): starting preview");
    try{
      seq1 = MidiSystem.getSequencer();
      if(seq1 == null) {
        return; // throw exception here in the future
      } else {
        // get resources to get midi
        seq1.open();
      }
      // sequencer gets file
      seq1.setSequence(sequenceNew);
      Thread t = new Thread(new Runnable() {
        public void run() {
          seq1.start();
        }
      });
      t.start();
    } catch(Exception ex){
      ex.printStackTrace();
    }
  }

  public boolean isPlaying() {
    return seq1.isRunning();
  }

  public void saveFile(String filename) {
    logger.log("Humanizer: saveFile(filename): saving MIDI file");
    if (sequenceNew == null) return; // throw exception here in the future
    try {
      MidiSystem.write(sequenceNew, 1, new File(filename));
    } catch (Exception ex) {
      ex.printStackTrace();
    }
  }

  private void requestSequence() {
    logger.log("Humanizer: requestSequence(): requesting MIDI sequence");
    MidiFileHandler mhandle = MidiFileHandler.getInstance();
    if (!mhandle.loadMidiFile(this.midiFile))
      // throw exception here in the future
      return;
    sequenceOrig = mhandle.getMidiSequence();
  }

  public boolean humanizationCompleted() {
    return passescompleted == passcount;
  }

  private void modifySequence() {
    for(int i = 0; i < 5; i++) { // try to humanize up to 5 times
      logger.log("Humanizer: modifySequence(): modifying MIDI sequence");
      initialized = false;
      requestSequence();
      initialized = true;
      sequenceNew = sequenceOrig; // copy the original sequence
      passcount = profiles.size();
      passescompleted = 0;
      for (String s : profiles.keySet()) {
        modifySequence(s, profiles.get(s));
        passescompleted++;
      }
      if (validateResults()) break;
    }
  }


  private void modifySequence(String selection, File profile) {
    // string in the form of: ##X (where ## is a channel number, and X is a parameter)

    String attr = selection.substring(selection.length()-1).toUpperCase();
    int trackNum = Integer.parseInt(selection.substring(0,selection.length()-1))-1;

    boolean isGaussian = false;

    if (profile == null) return;
    ProfileHandler.setProfile(profile);
    if (ProfileHandler.isDiscrete()) {
      HashMap<Double,Double> discrete = ProfileHandler.getDiscrete();
      DiscreteRNG.setProbabilities(discrete);
    } else if (ProfileHandler.isGaussian()) {
      gaussian = ProfileHandler.getGaussian();
      isGaussian = true;
    } else return;


    // requires Java 7 (faster, but uglier)
    switch (attr) {
      case "A":
        modifyAmplitude(trackNum, profile, isGaussian);
        break;
      case "T":
        modifyTiming(trackNum, profile, isGaussian);
        break;
      case "V":
        modifyVelocity(trackNum, profile, isGaussian);
        break;
      default:
        // throw exception here in the future
        return;
    }

    // non-Java 7 version (slower, cleaner, portable)
    /*
    if (attr.equalsIgnoreCase("A"))
      modifyAmplitude(trackNum, profile);
    else if (attr.equalsIgnoreCase("T"))
      modifyTiming(trackNum,profile);
    else if (attr.equalsIgnoreCase("V"))
      modifyVelocity(trackNum, profile);
    else
      // throw exception here in the future
      return;
    */
  }

  // MIDI Messages of interest:
  // XXXX is channel #
  // KKKKKKK is note #
  // VVVVVVV is velocity
  // PPPPPPP is key pressure
  // 1001XXXX 0KKKKKKK 0VVVVVVV -> Note On (for Velocity)
  // 1010XXXX 0KKKKKKK 0PPPPPPP -> Polyphonic Key Pressure (for Amplitude)
  // for timing, manipulate MidiEvent.setTick() directly

  private void modifyVelocity(int tracknum, File profile, boolean isGaussian) {
    logger.log("Humanizer: modifyVelocity(tracknum,profile,isGaussian): modifying channel " +
tracknum + " with profile " + profile.getName());
    Track[] oldTracks = sequenceNew.getTracks();
    Track[] newTracks = new Track[oldTracks.length];
    for (int i = 0; i < oldTracks.length; i++) {
      Track trackold = oldTracks[i];
      sequenceNew.deleteTrack(trackold);
      newTracks[i] = sequenceNew.createTrack();
      int numevents = trackold.size();
      for (int ii = 0; ii < numevents; ii++) {
        MidiEvent event = trackold.get(ii);
        MidiMessage message = event.getMessage();
        int statusbyte = message.getStatus() >>> 4;
        //System.out.println("statusbyte: " + statusbyte + " velomask: " + VELO_MASK);
        if (statusbyte != VELO_MASK) { // not a velocity command
          newTracks[i].add(event);
        }
        else { // a velocity command
          if((statusbyte & 0b1111) == tracknum) {
            logger.log("   -> velocity event!");
            byte[] data = message.getMessage();
            if (isGaussian) {
              Double[] p = ProfileHandler.getGaussian();
              data[2] += (int)gaussianRNG.next(p[0],p[1]);
            } else {
              data[2] += (int)discreteRNG.next();
            }
            data[2] &= ˜(0b10000000); // constrain values
            long time = event.getTick();
            //System.out.println("time: " + time);
            //System.out.println("data: " + data[0] + " " + data[1] + " " + data[2]);


            try {
              event = new MidiEvent(new ShortMessage(data[0],data[1],data[2]),time);
            } catch (Exception ex) {ex.printStackTrace();}

          }
          newTracks[i].add(event);
        }
      }
    }
  }

  private void modifyAmplitude(int tracknum, File profile, boolean isGaussian) {
    logger.log("Humanizer: modifyAmplitude(tracknum,profile,isGaussian): modifying channel " +
 tracknum + " with profile " + profile.getName());
    Track[] oldTracks = sequenceNew.getTracks();
    Track[] newTracks = new Track[oldTracks.length];
    for (int i = 0; i < oldTracks.length; i++) {
      Track trackold = oldTracks[i];
      sequenceNew.deleteTrack(trackold);
      newTracks[i] = sequenceNew.createTrack();
      int numevents = trackold.size();
      for (int ii = 0; ii < numevents; ii++) {
        MidiEvent event = trackold.get(ii);
        MidiMessage message = event.getMessage();
        int statusbyte = message.getStatus() >>> 4;
        //System.out.println("statusbyte: " + statusbyte + " ampmask: " + AMP_MASK);
        if (statusbyte != AMP_MASK) { // not an amplitude command
          newTracks[i].add(event);
        }
        else { // an amplitude command
          if((statusbyte & 0b1111) == tracknum) {
            logger.log("   -> amplitude event!");
            byte[] data = message.getMessage();
            if (isGaussian) {
              Double[] p = ProfileHandler.getGaussian();
              data[2] += (int)gaussianRNG.next(p[0],p[1]);
            } else {
              data[2] += (int)discreteRNG.next();
            }
            data[2] &= ˜(0b10000000); // constrain values
            long time = event.getTick();
            //System.out.println("time: " + time);
            //System.out.println("data: " + data[0] + " " + data[1] + " " + data[2]);


            try {
              event = new MidiEvent(new ShortMessage(data[0],data[1],data[2]),time);
            } catch (Exception ex) {ex.printStackTrace();}

          }
          newTracks[i].add(event);
        }
      }
    }
  }

  private void modifyTiming(int tracknum, File profile, boolean isGaussian) {
    logger.log("Humanizer: modifyTiming(tracknum,profile,isGaussian): modifying channel " + tr
acknum + " with profile " + profile.getName());
    Track[] oldTracks = sequenceNew.getTracks();
    Track[] newTracks = new Track[oldTracks.length];
    for (int i = 0; i < oldTracks.length; i++) {
      Track trackold = oldTracks[i];
      sequenceNew.deleteTrack(trackold);
      newTracks[i] = sequenceNew.createTrack();
      int numevents = trackold.size();
      for (int ii = 0; ii < numevents; ii++) {
        MidiEvent event = trackold.get(ii);
        MidiMessage message = event.getMessage();
        int statusbyte = message.getStatus() >>> 4;
        //System.out.println("statusbyte: " + statusbyte + " timemask: " + TIME_MASK);
        if (statusbyte != TIME_MASK) { // not a timing command
          newTracks[i].add(event);
        }
        else { // a timing command
          if((statusbyte & 0b1111) == tracknum) {
            logger.log("   -> timing event!");
            long time = event.getTick();
            if (isGaussian) {
              Double[] p = ProfileHandler.getGaussian();
              time += (long)gaussianRNG.next(p[0],p[1]);
            } else {
              time += (long)discreteRNG.next();
            }
            //System.out.println("time: " + time);
            //System.out.println("data: " + data[0] + " " + data[1] + " " + data[2]);


            try {
              event = new MidiEvent(message,time);
            } catch (Exception ex) {ex.printStackTrace();}

          }
          newTracks[i].add(event);
        }
      }
    }
  }

  private boolean validateResults() {
    logger.log("Humanizer: validateResults(): validating results...");
    long threshold = 5000000; // +/- 5 seconds
    boolean retval = false;
    if (Math.abs(sequenceOrig.getMicrosecondLength() -
                 sequenceNew.getMicrosecondLength()) <= threshold)
      retval = true;
    logger.log("   -> " + retval);
    return retval;
  }
```

```
}
```

```
#!/bin/bash

CLASSPATH="."
BUILDDIR="../classes/"

echo "building all:"
sh build-gui.sh $1
sh build-io.sh $1
sh build-statistics.sh $1
sh build-processes.sh $1

echo "build complete."
```

```
#!/bin/bash

CLASSPATH="."
BUILDDIR="../classes/"

cd src/
echo "Building ladasha.GUI..."
javac -cp $CLASSPATH -d $BUILDDIR $1 ladasha/GUI/*.java
```

```
#!/bin/bash

CLASSPATH="."
BUILDDIR="../classes/"

cd src/
echo "Building ladasha.io..."
javac -cp $CLASSPATH -d $BUILDDIR $1 ladasha/io/*.java
```

```
#!/bin/bash

CLASSPATH="."
BUILDDIR="../classes/"

cd src/
echo "Building ladasha.processes..."
javac -cp $CLASSPATH -d $BUILDDIR $1 ladasha/processes/*.java
```

```
#!/bin/bash

CLASSPATH="."
BUILDDIR="../classes/"

cd src/
echo "Building ladasha.statistics..."
javac -cp $CLASSPATH -d $BUILDDIR $1 ladasha/statistics/*.java
```

```
#!/bin/bash

rm -rf *-new.mid classes/ladasha humanizer-*.tar.gz *~ tex/*.aux tex/*.log tex/*.synctex.
gz tex/*.toc classes/*.log classes/*.jar
```

```
#!/bin/bash

cd classes
java ladasha.GUI.GUI
```

```
#!/bin/bash

./clean.sh
./build.sh $1
./run.sh
```

```
#!/bin/bash

NOW=‘date +"%Y%m%d"‘
TAR=‘which tar‘

echo "Building humanizer-$NOW.tar.gz..."
$TAR -cvzf humanizer-$NOW.tar.gz ./*
```

```
#!/bin/bash

echo "Making jar..."
cd classes/
jar -cvfm midihumanizer.jar META-INF/MANIFEST.MF ./ladasha/
echo "done!"
```

```
#!/bin/bash

cd classes/
java -jar midihumanizer.jar
```

```
MIDI Humanizer Changelog:
Team La-a

Key:
+ new feature
- removed feature
!! bug report

=========================
01Nov2011:
  + made initial build system
  + added core functionality to the file import pane

02Nov2011:
  + cleaned up the build system (made POSIX compliant)
  + added maketar.sh to build system
  + added this changelog
  + refactored GUI - each card has a set of initialization tasks

03Nov2011:
  + added in Jason’s RNG code (draft)
  + converted Jason’s RNG code to singletons
  + documented the GUI code...in comments
  + changed Jason’s variable names... hah
  + added in Jamie’s MidiFileHandler (draft)

04Nov2011:
  + updated Jamie’s MidiFileHandler
  + updated the GUI (it looks professional-ish now)
  + added functionality to the preview button
  + commited build alpha2 to the wiki

05Nov2011:
  + changed preview button to play/stop style
  + added some graphical changes
  + recommited the new build as build alpha2

08Nov2011:
  + began implementing profile selection panel
  !! noticed bug (priority: medium): no guards on card progression
     -> example: no loaded midi file, allowed to continue to profile selection
     -> example: no loaded profiles, allowed to continue to parameter selection
  !! noticed bug (priority: low): no watch on preview/stop button
     -> example: preview whole file, button stays as a stop button
  + allowed multiple profiles to be loaded (one at a time)
  + JLabel list to show loaded profiles
  + initial help (welcome, load file, load profiles)

10Nov2011:
  + rearranged profile selection panel
  + fixed a small nullpointerexception on midi file loads
  + added a clear selected profiles button
  + added moar cowbell
  + worked on Jamie’s ProfileHandler code

15Nov2011:
  + Worked on ProfileHandler
  + Worked on Humanizer
  + Worked on GUI
  + Refactored build system
  + What has been seen cannot be unseen

16Nov2011:
  + Humanizer is pretty close to finished
  + Set GUI modifications assignment card initial format
  + Retooled build system again (minor)
  + A few "default" profiles included
  + Fixed a few [rawtypes] warnings

17Nov2011:
  + Added "All" to selection of tracks and parameters for modification
  + Added comments (recursively) to GUI code
  + Eat at Joe's
  + Next button now disabled until selections made
  + Cleaned code a bit
  + Wrote a test case
  + ProfileHandler updated

18Nov2011:
  + Refactored GaussianRNG and DiscreteRNG
  + Refactored ProfileHandler
  + Refactored MidiFileHandler
  + Implemented first stages of Humanizer
  + Added preview card and file save card

19Nov2011:
  + So, it works now.
  + Humanizer finished
  + Refactored DiscreteRNG to take in a hashmap instead of two arraylists
  + Reverted to 16 tracks in tracks JComboBox
  + Relabeled tracks -> channels in GUI
  + Fixed minor bugs
  + Added a handful of gaussian profiles
  !! noticed bug (priority: high) data2 bytes out of range: exceptions [affects: humanize
r]
  !! noticed bug (priority: high) threshold not checked to validate changes [affects: GUI
]

20Nov2011:
  + Fixed regression: DiscreteRNG works properly now
  + Fixed bug: data2 bytes out of range
  + Added Dis3.mhp profile

21Nov2011:
  + Fixed bug: loading custom profiles now functions properly
  + Added function to create custom gaussian profiles from GUI
  + Added time-keeping JProgressBars to preview cards
  + Implemented logger

22Nov2011:
  + Ran through findbugs: squashed 'em
  + Fixed too many files open error (closing scanners in the ProfileHandler)
  + Fixed humanizer not working... by taking it out of the thread. :(
  !! noticed bug (priority: medium) sequence re-humanized when returning from help pane

27Nov2011:
  + Added a makejar.sh and runjar.sh to the build system

28Nov2011:
  + Refactored humanizer to support progress bar in the future
  + Humanizer now tries to humanize up to five times before bailing on un-met threshold
  + Minor change to the clean.sh - now deletes all files matching the pattern *-new.mid

29Nov2011:
  + Implemented progress bar for humanization
  + Cleaned up background processes in GUI a bit
  + Fixed re-humanization bug noticed on 22Nov
  + Added JOptionPane to indicate successful saving of profile and midi files
  + Fixed (hopefully) JComboBox appearance on modifications selection card

30Nov2011:
  + Merged Jason's changes
```

```
1,.01,2,.02,3,.03,4,.04,5,.05,6,.06,7,.07,8,.08,9,.09,10,.1,11,.11,12,.12,13,.13,14,.09
```

```
1,.09,2,.13,3,.12,4,.11,5,.1,6,.09,7,.08,8,.07,9,.06,10,.05,11,.04,12,.03,13,.02,14,.01
```

```
1,.02,2,.02,3,.02,4,.03,5,.03,6,.04,7,.04,8,.05,9,.05,10,.05,11,.05,12,.05,13,.05,14,.05,
15,.05,16,.05,17,.05,18,.05,19,.05,20,.05,21,.01,22,.01,23,.01,24,.01,25,.01,26,.01,27,.0
1,28,.01,29,.01,30,.01,31,.01,32,.01,33,.01,34,.01,35,.01
```

```
[0,15]
```

```
[0,125]
```

```
[0,40]
```

```
[0,7]
```

```
[0,25]
```

```
[0,1]
```

```
[0,2]
```

```
[0,10000]
```

```
[0,3]
```

6 Project Analysis

MIDI Humanizer: Project Analysis

Team La-a

Fall, 2011

Contents

1 Hours worked on various components of project

1.1 DiscreteRNG

Jason spent 2 hours working on this component.
Jake spent 10 minutes working on this component.

1.2 GaussianRNG

Jason spent 1 hour working on this component.
Jake spent 10 minutes working on this component.

1.3 MidiFileHandler

Jamie spent 2 hours working on this component.
Jason spent 30 minutes working on this component.
Jake spent 20 minutes working on this component.

1.4 ProfileHandler

Jamie spent 2 hours working on this component.
Jason spent 45 minutes working on this component.
Jake 10 minutes working on this component.

1.5 Logger

Jake spent 2 hours working on this component.

1.6 Humanizer

Jake spent 2.5 hours working on this component.
Jason spent 15 minutes working on this component.

1.7 GUI

Jake spent 5 hours working on this component.
Jason spent 30 hours (!) working on this component.

1.8 build system

Jake spent 1 hour working on this component.

2 Metric: Lines of Code/Hour

2.1 Team as a whole

Total lines of code: 2137
Total hours spent: 49.83
Lines of Code/Hour: 42.88

2.2 Jason

Estimated lines of code: 1037
Jason's hours spent: 34.5
Lines of Code/Hour: 30.06

2.3 Jamie

Estimated lines of code: 273
Jamie's hours spent: 4
Lines of Code/Hour: 68.25

2.4 Jake

Estimated lines of code: 887
Jake's hours spent: 11.34
Lines of Code/Hour: 78.22

7 Reflections on Process

Reflection: Jacob Peck

Working on the MIDI Humanizer project has been very interesting and an exhilarating experience. The fact that we were able to come together as a team and complete the project within 15 weeks is nothing short of astounding to me. Jason and Jamie have awesome work ethic, and they really kicked me into shape.

I learned a lot about larger-scale software packages and applications, and how long they truly take to code. As a mostly independent coder, I never realized just how long I spend writing code, testing, reiterating, etc. The process really is a grind, when you look at it, and it becomes more of a time vampire than you would expect, just from looking at the process from the outside. Indeed, most of the time, I found myself thinking “this shouldn’t take too long to implement…” and end up not finishing it until the next week.

MIDI Humanizer is a multi-faceted project, and a very viable solution to a rather niche problem. Our result works, and it works well. But that’s not to say that I wouldn’t have done things differently if I were implementing this myself. I would have made this a command-line application with pipeable output, but the team as a whole was more comfortable using a GUI to guide user input. I would have also implemented the Logger much sooner than I did, as it proved valuable in tracking down bugs. But, hindsight is 20/20, as they say. Overall, I had a good time, learned a lot, and made some new friends. Success, I’d say.

Going into the project, I didn't really know what to expect, as I had yet to work on a non-trivial program, and had not collaborated with others in this respect. The planning of the project and really thinking things through prior to coding really made it easy to put everything together and make it work. I was very anxious to start the process for weeks before we actually got started, but I think there would have been a lot of wasted effort had we not had a solid grasp of what we were to build. The project could have gone in a number of directions, but we were able to easily decide on virtually every aspect. The team kept a rigid schedule and we were able to meet all deadlines, due to a well thought out process and format. Overall I am very pleased with the progression, and all the care that the team took to see the production succeed.

My main focus at the start was handling the algorithms to randomize the notes in a midi file. This didn't really take too long, and I intended to write several different algorithms, but in the process I realized that a gaussian distribution best simulates human error, and that a "discrete" random generator could be used to provide custom alterations. We wanted the user to be able to customize their humanizations, so we decided to allow for input files, or "profiles" to be loaded into the program and used. I wrote a few of the profiles, as to provide some standard implementations, and even included a way for users to create their own custom gaussian profiles in the program.

Once the algorithms were in place, I took on the load of programming the user interface. Jake had provided a framework, based on our collective decision to use a "wizard" type interface, and I build upon that. I created custom graphics, and pretty much built the interface from there. I am very much a perfectionist, so it took quite a while to get it just right, I think of the @600 lines of code I put into the GUI, I averaged maybe 20 lines per hour, but, I think they were very high quality lines, and I was more than willing to take the extra time to tweak it just right.

I really enjoyed working on this project, I learned a ton about GUI's, and quite a bit about pseudo-random algorithms. I learned a lot about tweaking code to perfection and different strategies for making code easy to maneuver, considering the vast amounts of debugging and tweaking we were doing. I also learned a lot from working with my group, as our leader is vastly more experienced than I and was willing to take the time to explain the "nitty gritty."

Jason Shaffner

In CSC 380 skills, Team work, communicate, documentation, UML mapping, how a business in the real world really works, what customers and clients look for in programs, how to budget programs.

During the semester I learned a lot in a team then I would have learned by my self. When working on the MIDI humanizer Jake showed us how to build a tarball and showed and helped us to coordinate our work via wiki. Jake also showed and explained to me what a hash map was, so wells explained, that I was able to use and master the hash map when I left the room.

I Some team work skills that I also worked on was communication and criticism. During the project I improve my communication skills by telling my problems with my team and getting them ironed out one at a time. One example of communication and adapting is my first attempt at the Midi file loader. When I first coded this program I thought it was short but sweet and accomplished what the team needed. Later I learned that I had to adapt my program to meet the changing requirements for our program. When I showed my team my code I was given good advice and improved my program significantly. Thought I did feel a little disheartened when I herd my program does nothing it needed to so then I was happy when I learned how to fix it and improved it. This event also helped me to improve my skills in java and in general as a programmer.

Documentation was also a very important issue in team la-a. If out code was not document all of our parts of out program could not have functioned at all. But at the same time I learned that if I over documented on things I was just wasting time. With this assignment I learned how to improved my current standards of documentation and was able to effectively use my code in the program. An example of this was when I was working on the profile handler. At first I was over documenting all of my code and now I am making my comments better and more precise. Also the test cases that I worked on helped me improve my documentation a skill. I learned what goes into test cases and how much thought. In conclusion I learned a lot form my team and learned how to become a better programmer in general. We covered all the topics I hit on and much more. I think that this class was very beneficial to my programming career and will continue to help me down the road.