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.

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.