



TODO-Team-Name

MEMBERS: RYAN LILE, CURTIS MOTES, PAIGE PECK

PROJECT: JYTHON MUSIC

What is JythonMusic?

- ▶ A free open source environment that allows users to create music through python programming
- ▶ An extension of the jMusic library, which provides composition and audio tools to programmers in Java
- ▶ Written in Python

JythonMusic
Creative Programming in Python



JythonMusic Structure

- ▶ JEM.jar executable file used to run JythonMusic environment
- ▶ Library Folder containing core python files:
 - ▶ Music library
 - ▶ GUI and Image library
 - ▶ Timer library
 - ▶ MIDI library
 - ▶ OSC(Open Sound Control) library

Automated Testing Framework

- ▶ Invoked by bash script 'RunAllTests.sh' at top-level directory
- ▶ Clears any existing test output
- ▶ Accesses testCases folder within TestAutomation directory
- ▶ Uses .txt files within testCases to populate array for entry into HTML table
- ▶ Uses .txt files to supply inputs to python tests
- ▶ Runs JEM, executes test, pipes output to HTML table
- ▶ Compares oracle to actual output to determine Pass/Fail

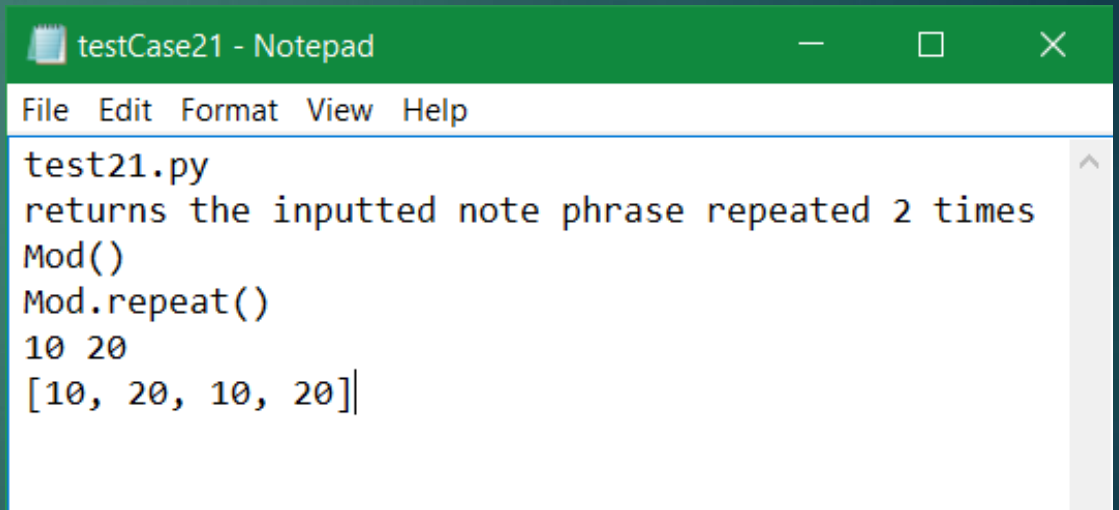
Note: No Spaces in any folder or file names

```
/TestAutomation
  /project
    /src
    /bin
    / ...
  /scripts
    runAllTests.(some scripting extension)
    ... other helper scripts
  /testCases
    testCase1.txt
    testCase2.txt
    ...
  /testCasesExecutables
    testCase1(may be folder or file)
    ...
  /temp (for output from running tests ... be sure to clean at start of runAllTests)
    testCase1results (might be folder or file)
    ...
  /oracles
    testCase1Oracle (might be folder or file)
    ...
  /docs
    README.txt
  /reports
    testReport.(txt or html)
```

Test Case Format

- ▶ Test Cases are formatted in the following way, which each property on a new line within the text file:

1. Test number
2. Requirement being tested
3. Component being tested
4. Method being tested
5. Test Input(s)
6. Expected output (Oracle)



```
test21.py
returns the inputted note phrase repeated 2 times
Mod()
Mod.repeat()
10 20
[10, 20, 10, 20]
```

Components/Methods Being Tested

In music.py

- ▶ Note:
 - ▶ getPitch()
 - ▶ setPitch()
- ▶ Phrase:
 - ▶ addNote()
 - ▶ getNoteList()
 - ▶ empty()
 - ▶ copy()

▶ Mod:

- ▶ repeat(phrase)
- ▶ palindrome(phrase)

In timer.py

▶ Timer2:

- ▶ isRunning()
- ▶ setRepeat()
- ▶ start()
- ▶ stop()
- ▶ getDelay()

Running the Framework

- ▶ The testing framework is invoked by using the command `./scripts/runAllTests.sh` from within the TestAutomation directory. Here is how it looks while executing:

```
paige@paige-VirtualBox:~/TODO-Team-Name/TestAutomation$ ./scripts/runAllTests.sh  
Clearing Temp Folder  
Running Tests  
  
Running Test 01  
Running Test 02  
Running Test 03  
Running Test 04  
Running Test 05  
Running Test 06  
Running Test 07  
Running Test 08  
Running Test 09  
Running Test 10  
Running Test 11  
Running Test 12  
Running Test 13  
Running Test 14
```


Testing Report

- ▶ Once tests are finished running, the testing framework produces an HTML report with CSS styling displaying test case information and pass/fail result of test.
- ▶ Report opens in a new browser window using the default browser.
- ▶ Pass/Fail is generated by comparing piped output from test.py files to oracle in txt file.

Passing Tests

Test	Requirement	Component	Method	Input	Expected Output	Actual Output	Result
test1.py	returns true when instance is running	Timer2	Timer2.start		True	True	Pass
test2.py	checks that if the flag is true, repeat is also true	Timer2	Timer2.setRepeat(flag)		True	True	Pass
test3.py	timer task starts on command	Timer2	Time2.start()		True	True	Pass
test4.py	time object stops after being created and started	Timer2	timer2.stop()		False	False	Pass
test5.py	ability to use delay time ionterval	Timer2	Tiemr2.timeInterval()	1000	1000	1000	Pass
test6.py	returns note that was added to phrase	Phrase	Phrase.addNote(pitch, rhythmValue)	1	[1]	[1]	Pass
test7.py	returns 2 notes that were added to phrase	Phrase	Phrase.addNote(pitch, rhythmValue)	1 60	[1, 60]	[1, 60]	Pass
test8.py	returns note that was added to phrase	Phrase	Phrase.addNote(pitch, rhythmValue)	127	[127]	[127]	Pass
test9.py	returns size of phrase after note fails to be added to phrase	Phrase	Phrase.addNote(pitch, rhythmValue)	150	0	1	Fail
test10.py	returns size of phrase after note is added	Phrase	Phrase.addNote(note)	10 20 30 40	4	4	Pass
test11.py	returns size of phrase after emptying note list	Phrase	Phrase.empty()		0	0	Pass
test12.py	returns true if the copied phrase matches original	Phrase	Phrase.copy()		True	True	Pass
test13.py	returns true if the endTime of the note list matches expected duration	Phrase	Phrase.addNote()		True	True	Pass
test14.py	returns true if the duration of the phrase is 10 after adding 4 notes totaling 10 duration	Phrase	Phrase.getEndTime()		True	True	Pass
test15.py	returns true if two phrases match, using a list for one phrase, and individually adding notes to another	Phrase	Phrase.addNoteList(pitchList, rhythmList)		True	True	Pass
test16.py	returns note list of chord added	Phrase	Phrase.addChord(pitchList, rhythm)	1 2 3	[1, 2, 3]	[1, 2, 3]	Pass
test17.py	returns pitch of note (entered as pitch constant) as MIDI integer 0-127	Note	Note.getPitch()	0	0	0	Pass
test18.py	returns pitch of note (entered as pitch constant) as MIDI integer 0-127	Note	Note.getPitch()	1	1	1	Pass
test19.py	returns pitch of note (entered as pitch constant) as MIDI integer 0-127	Note	Note.getPitch()	127	127	127	Pass
test20.py	returns pitch of note (entered as pitch constant) as MIDI integer 0-127	Note	Note.getPitch()	64	64	64	Pass
test21.py	returns the inputted note phrase repeated 2 times	Mod	Mod.repeat()	10 20	[10, 20, 10, 20]	[10, 20, 10, 20]	Pass
test22.py	returns a reverse-order note phrase appended to original phrase	Mod	Mod.palindrome(phrase)	127	[127, 127]	[127, 127]	Pass
test23.py	returns a reverse-order note phrase appended to original phrase	Mod	Mod.palindrome(phrase)		[]	[]	Pass
test24.py	returns a reverse-order not phrase appended tooriginal phrase	Mod	Mod.palindrome(phrase)	40 60	[40, 60, 60, 40]	[40, 60, 60, 40]	Pass
test25.py	returns a reverse-order note phrase appended to original phrase	Mod	Mod.palindrome(phrase)	40 60 80	[40, 60, 80, 80, 60, 40]	[40, 60, 80, 80, 60, 40]	Pass

Fault-Injection

Fault-injection is the process of intentionally altering logic within the source code of tested files to check whether tests are properly testing requirements of individual methods.

We injected 5 faults into our source code at different locations within each component being tested.

Fault Injection Template:

Fault template:

Location of file

The line number that was changed, the class, and the method

Original line

Fault injected line

Tests that will fail

Example:

Fault 2)

TODO-Team_Name/TestAutomation/project/Jython/library/timer.py

Line 320, Timer2 class, start method

if self._running == True

if self._running == False

Test 1 fails

Fault-Injection Results

Test	Requirement	Component	Method	Input	Expected Output	Actual Output	Result
test1.py	returns true when instance is running	Timer2	Timer2.start		True	False	Fail
test2.py	checks that if the flag is true, repeat is also true	Timer2	Timer2.setRepeat(flag)		True	Audio: AudioDevice: PulseAudio Mixer, max in = 6, max out = 6 Audio: AudioDevice: default [default], max in = 32, max out = 32 Audio: AudioDevice: I82801AAICH [plughw:0,0], max in = 2, max out = 0 Audio: AudioDevice: I82801AAICH [plughw:0,1], max in = 2, max out = 0 Audio: AudioDevice: Port I82801AAICH [hw:0], max in = 0, max out = 0 ---- Pure Java JSyn www.softsynth.com - rate = 44100, RT, V16.5.14 (build 448, 2012-12-10) Traceback (most recent call last): File "library/test2.py", line 9, in t.setRepeat(flag) File "/home/paige/TODO-Team-Name/TestAutomation/project/Jython/library/timer.py", line 283, in setRepeat self._repeat = !flag # Fault Injection 3 NameError: global name 'lag' is not defined Output buffer size = 7056 bytes. Output Latency = 40.0 msec	Fail
test3.py	timer task starts on command	Timer2	Time2.start()		True	True	Pass
test4.py	time object stops after being created and started	Timer2	timer2.stop()		False	False	Pass
test5.py	ability to use delay time on interval	Timer2	Tiemr2.timeInterval()	1000	1000	0	Fail
test6.py	returns note that was added to phrase	Phrase	Phrase.addNote(pitch, rhythmValue)	1	[1]	[1]	Pass
test7.py	returns 2 notes that were added to phrase	Phrase	Phrase.addNote(pitch, rhythmValue)	1 60	[1, 60]	[1, 60]	Pass
test8.py	returns note that was added to phrase	Phrase	Phrase.addNote(pitch, rhythmValue)	127	[127]	[127]	Pass
test9.py	returns size of phrase after note fails to be added to phrase	Phrase	Phrase.addNote(pitch, rhythmValue)	150	0	1	Fail
test10.py	returns size of phrase after note is added	Phrase	Phrase.addNote(note)	10 20 30 40	4	Audio: AudioDevice: PulseAudio Mixer, max in = 6, max out = 6 Audio: AudioDevice: default [default], max in = 32, max out = 32 Audio: AudioDevice: I82801AAICH [plughw:0,0], max in = 2, max out = 0 Audio: AudioDevice: I82801AAICH [plughw:0,1], max in = 2, max out = 0 Audio: AudioDevice: Port I82801AAICH [hw:0], max in = 0, max out = 0 ---- Pure Java JSyn www.softsynth.com - rate = 44100, RT, V16.5.14 (build 448, 2012-12-10) Traceback (most recent call last): File "library/test10.py", line 6, in note1 = Note(int(inary[0]),1) File "/home/paige/TODO-Team-Name/TestAutomation/project/Jython/library/music.py", line 1027, in __init__ raise TypeError("Note pitch should be an integer between 0 and 127 (it was " + str(value) + ").") TypeError: Note pitch should be an integer between 0 and 127 (it was 10).Output buffer size = 7056 bytes. Output Latency = 40.0 msec	Fail
test11.py	returns size of phrase after emptying note list	Phrase	Phrase.empty()		0	Audio: AudioDevice: PulseAudio Mixer, max in = 6, max out = 6 Audio: AudioDevice: default [default], max in = 32, max out = 32 Audio: AudioDevice: I82801AAICH [plughw:0,0], max in = 2, max out = 0 Audio: AudioDevice: I82801AAICH [plughw:0,1], max in = 2, max out = 0 Audio: AudioDevice: Port I82801AAICH [hw:0], max in = 0, max out = 0 ---- Pure Java JSyn www.softsynth.com - rate = 44100, RT, V16.5.14 (build 448, 2012-12-10) Output buffer size = 7056 bytes. Output Latency = 40.0Traceback (most recent call last): File "library/test11.py", line 7, in note1 = Note(10,1) File "/home/paige/TODO-Team-Name/TestAutomation/project/Jython/library/music.py", line 1027, in __init__ raise TypeError("Note pitch should be an integer between 0 and 127 (it was " + str(value) + ").") TypeError: Note pitch should be an integer between 0 and 127 (it was 10). msec	Fail
test12.py	returns true if the copied phrase matches original	Phrase	Phrase.copy()		True	Audio: AudioDevice: PulseAudio Mixer, max in = 6, max out = 6 Audio: AudioDevice: default [default], max in = 32, max out = 32 Audio: AudioDevice: I82801AAICH [plughw:0,0], max in = 2, max out = 0 Audio: AudioDevice: I82801AAICH [plughw:0,1], max in = 2, max out = 0 Audio: AudioDevice: Port I82801AAICH [hw:0], max in = 0, max out = 0 ---- Pure Java JSyn www.softsynth.com - rate = 44100, RT, V16.5.14 (build 448, 2012-12-10) Output buffer size = 7056 bytes. Output Latency = Traceback (most recent call last): File "library/test12.py", line 7, in note1 = Note(10,1) File "/home/paige/TODO-Team-Name/TestAutomation/project/Jython/library/music.py", line 1027, in __init__ raise TypeError("Note pitch should be an integer between 0 and 127 (it was " + str(value) + ").") 40.0 msecTypeError: Note pitch should be an integer between 0 and 127 (it was 10).	Fail

Overview/Works Cited

- ▶ GitHub: <https://github.com/CSCI-362-01-2016/TODO-Team-Name>
- ▶ JythonMusic: <https://jythonmusic.me/>
- ▶ Beginner Bash: <http://www.tldp.org/LDP/Bash-Beginners-Guide/html/>