CS100J Spring 2008 Assignment A6. Mozart's Musikalisches Würfelspiel. Due 11:59PM, Saturday, 12 Apr

This assignment uses two-dimensional arrays and random-number generation in an interesting setting. Inspiration for it comes from an assignment given by Kevin Wayne and Robert Sedgewick in Computer Science at Princeton. We have used this assignment before in CS100J, and, although we have made a few modifications, it is easy to cheat. We ask you not to do so. You won't learn anything that way, we have to do extra unnecessary grading, and if you are caught it makes more work for all of us.

Please keep track of the time you spend on this assignment. At the end, we ask you to put a comment at the top of file `WurfelSpiel.java` that indicates how much time you spent.

In 1787, Wolfgang Amadeus Mozart created a dice game (Mozart's Musikalisches Würfelspiel). One composes a two-part waltz by pasting together 32 of 272 pre-composed musical elements at random. The waltz consists of two parts: a minuet and a trio. Each is composed of 16 measures, which are generated at random according to a fixed set of rules, as described below.

*Minuet*. The minuet consists of 16 measures. The 176 possible Minuet measures are named `m1.wav` through `m176.wav`. To determine which one to play, roll two dice and use the following table. The dice roll (a row number) tells you what to play for a measure (a column number). For example, if you roll 11 for measure 3, then play wav file m165.wav.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 96 | 22 | 141 | 41 | 105 | 122 | 11 | 30 | 70 | 121 | 26 | 9 | 112 | 49 | 109 | 14 |
| 3 | 32 | 6 | 128 | 63 | 146 | 46 | 134 | 81 | 117 | 39 | 126 | 56 | 174 | 18 | 116 | 83 |
| 4 | 69 | 95 | 158 | 13 | 153 | 55 | 110 | 24 | 66 | 139 | 15 | 132 | 73 | 58 | 145 | 79 |
| 5 | 40 | 17 | 113 | 85 | 161 | 2 | 159 | 100 | 90 | 176 | 7 | 34 | 67 | 160 | 52 | 170 |
| 6 | 148 | 74 | 163 | 45 | 80 | 97 | 36 | 107 | 25 | 143 | 64 | 125 | 76 | 136 | 1 | 93 |
| 7 | 104 | 157 | 27 | 167 | 154 | 68 | 118 | 91 | 138 | 71 | 150 | 29 | 101 | 162 | 23 | 151 |
| 8 | 152 | 60 | 171 | 53 | 99 | 133 | 21 | 127 | 16 | 155 | 57 | 175 | 43 | 168 | 89 | 172 |
| 9 | 119 | 84 | 114 | 50 | 140 | 86 | 169 | 94 | 120 | 88 | 48 | 166 | 51 | 115 | 72 | 111 |
| 10 | 98 | 142 | 42 | 156 | 75 | 129 | 62 | 123 | 65 | 77 | 19 | 82 | 137 | 38 | 149 | 8 |
| 11 | 54 | 130 | 10 | 103 | 28 | 37 | 106 | 5 | 35 | 20 | 108 | 92 | 12 | 124 | 44 | 131 |
| 12 | 3 | 87 | 165 | 61 | 135 | 47 | 147 | 33 | 102 | 4 | 31 | 164 | 144 | 59 | 173 | 78 |

*Trio*. The trio consists of 16 measures. The 96 possible Trio measures are named `T1.wav` through `T96.wav`. To determine which one to play, roll one die and use the following table. For example, if you roll 1 for measure 21, then play file T81.wav.

| | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 72 | 6 | 59 | 25 | 81 | 41 | 89 | 13 | 36 | 5 | 46 | 79 | 30 | 95 | 19 | 66 |
| 2 | 56 | 82 | 42 | 74 | 14 | 7 | 26 | 71 | 76 | 20 | 64 | 84 | 8 | 35 | 47 | 88 |
| 3 | 75 | 39 | 54 | 1 | 65 | 43 | 15 | 80 | 9 | 34 | 93 | 48 | 69 | 58 | 90 | 21 |
| 4 | 40 | 73 | 16 | 68 | 29 | 55 | 2 | 61 | 22 | 67 | 49 | 77 | 57 | 87 | 33 | 10 |
| 5 | 83 | 3 | 28 | 53 | 37 | 17 | 44 | 70 | 63 | 85 | 32 | 96 | 12 | 23 | 50 | 91 |
| 6 | 18 | 45 | 62 | 38 | 4 | 27 | 52 | 94 | 11 | 92 | 24 | 86 | 51 | 60 | 78 | 31 |

**Example.** Here is a sample waltz generated using this process and the accompanying musical score. There are 11^16 * 6^16 different possible waltzes. Since this is over 10^23 different possibilities, each time you play the game you are likely to produce music that has never been heard before! Mozart carefully constructed the measures to obey a rigid harmonic structure, so each waltz reflects Mozart's distinct style.

**About this assignment.** To help you learn as much as possible in a short amount of time as possible, and also to give you advice on how to go about writing and testing programs, we lead you through this assignment in a series of steps. Please study the code we have given you; be conscious of style, specifications, etc.

Follow the directions carefully. Many people have had more points off because they failed to follow instructions than because of programming errors. When doing one step, don't go on to the next one until the current one is done and the method you wrote for it works!

**Step 1. Download** the following and place them all in a new folder for the assignment.
File StdAudio.java. Class `StdAudio` contains methods for manipulating and playing music in wave (.wav) format.
File WurfelSpiel.java. You will be writing class WurfelSpiel. We have provided a few components in it to help out.
File WurfelSpielTester.java. We have provided you with stubs of test methods and with some testing instructions in the bodies.
File measures.zip (34MB). After downloading this file, unzip it —into a folder `measures` that contains all the .wav files for the measures used in Mozart's Musikalisches Würfelspiel. Put folder `measures` in the folder with the two .java files.

Load `StudioAudio.java` into DrJava and become familiar with its methods. Call method `playScale` just to see what sounds come out.

**Step 2. Generating rolls of a die**. Your program will have to "roll a die" to produce a random number in the range 1..6. At the beginning of class `WurfelSpiel`, there is a declaration of a static `Random` variable `generator`. An object of class `generator` has functions for generating "random" numbers. The one you will use is function `nextInt`. Evaluation of a call `generator.nextInt(t)` yields an integer `i` that satisfies `0 <= i < t`. Use function `nextInt` in writing the body of method `WurfelSpiel.throwDie`. The method body need be only a single return statement.

Class `Random` is in API package `java.util`. Use link java.sun.com/javase/6/docs/api/java/util/Random.html to obtain the API specs for this class and spend some time becoming familiar with it.

Function `throwDie` seems extremely simple! Nevertheless, it has to be tested to make sure that (1) it produces only integers in the range 1..6 and (2) can produce all integers in 1..6. In order to help you do this, we have written part of method `WurfelSpielTester.testThrowDie`. Study what we have written in the specification and the body of this method, complete the method, and test `throwDie`.

**Step 3. A method for printing a String array.** Later, you will be writing a function that produces a `String` array of file names corresponding to measures to be played. In order to see that the method works, you have to see the array of `Strings`. For that purpose, we have partially written function `toString(String[])`. Study what we have written in the specification and function body and complete the function. Then test it by completing test method `WurfelSpielTester.testToString`.

**Step 4. Generating a waltz.** Before you work on creating a random waltz, first create a waltz assuming that each die thrown has the value 1, so that all the file names in row 2 of array `minuet` and row 1 of array `trio` are chosen. This allows you to concentrate on the idea of constructing a file name. To do this, write the body of function `WurfelSpiel.create1Spiel()`.

We have given you arrays `minuet` and `trio` in class `WurfelSpiel`. Each element of these arrays is an integer that indicates a file that represents a measure. For example, `minuet[2][1] = 96`, which represents file `measures/m96.wav`. So, you have to put the `String` `"measures/m96.wav"` into the array that this function returns. `minuet[2][1]` is one of the musical phrases that can be used in measure 1 of the minuet part. Note that "/" is used to separate folder name `measures` from file name `m96.wav`. Even if you have a PC, you must use "/" and not a backslash.

After writing this function, complete procedure `WurfelSpielTester.testCreate1Spiel()`. We put in a test to make sure that the first array element is correct so that you have some idea how to do this. You must make sure that all array elements are correct. You should also print the array of strings in the interactions pane, using function `toString(String[])`, so that you can see it; this latter step is just for your benefit.

**Step 5. Listen to the music!** Complete procedure `WurfelSpiel.play(String[])`, which will play all the files whose names are in an array such as that calculated by function `createSpiel`. You will notice pauses between measures when the music is played. We investigate eliminating the pauses later. Playing your array should produce the same music as this file: mozart12.wav.

**Step 6.** We can get rid of the pause between measures by building a single file that contains the music in the files given by an array like `s` in the past two steps. To do this, complete the body of function `WurfelSpiel.build(String)`.

To do this, you have to read a .wav file and place its contents into a **double** array; use function `StdAudio.read(String)` to read one file at a time. To help you out, we give an outline for the body of `WurfelSpiel.build(String)`, and we also give you method `WurfelSpiel.copy`. You figure out how to check this function; as long at it is correct, we will not look at your test cases.

**Step 7. Creating a Mozart Musikalisches Würfelspiel.** Finally, write the the body of function `WurfelSpiel.createRandomSpiel()`, which will create a random waltz. It should produce a random waltz as described at the beginning of this document. In this method, you can use `throwDie`, which you wrote earlier, to throw a die to get a number in the range 1..6. Also, use arrays `minuet` and `trio` in class `WurfelSpiel` when generating random names of files, as in method `create1Spiel`.

**Step 8. Saving a file.** You don't have to do anything here. We want you to know that method `StdAudio.save` allows you to save a **double** array in a .wav file, so that you can play it later ---or email it home to let your family know that you have been turned on by classical music and Musikalisches Würfelspiel. You can use any file name you want, as long as it ends in ".wav". The file will be stored on your desktop.

**Step 9. Submitting your assignment**. Place at the top of file `WurfelSpiel.java` a comment that says how many hours you spent on this assignment. Submit files `WurfelSpiel.java` and `WurfelSpielTester.java` on the CMS by the due date.