Project Name: Procedurally Generated Melody
Your Team Name: Musically Minded Modders
List of Team Members:

- Quinn A. Hannah-White
- Griffin Good
-

Our primary goal is to create a program that will write musical counterpoint on its own. Amateur composers starting out often work on contrapuntal exercises in order to help them understand underlying principles of music. Creating a counterpoint machine, then, is an important step on the path to automating the song writing process entirely. It will serve as a proof-of-concept as well as a trial run in order to get a sense of some of the obstacles and pitfalls. That being said, this will be slightly useful on its own, as three-lined counterpoint can often be used as choral music for use in more modern churches, especially counterpoint of the second or fourth species.

public class Driver

- Takes in input from the console, namely:
    - A string to determine the key
    - An integer to determine the species (between the values of 1 and 4 (so far))
    - An integer to determine the number of lines (1-3)
        - If 1, boolean for bass or melody
    - A boolean for compound
    - An integer to determine tempo (in beats per minute)
- Initializes the appropriate Species class
- Calls the assembleLines function
- Initializes the LineInterpreter
- Calls the toText and toMIDI functions from LineInterpreter

public enum Pitch

- Contains the values that Pitch can represent

public class Note

- Holds a Pitch
- Holds a Note, what comes after it
- Holds an integer, Begin (when the notes begin, measured in 12ths of a beat (based on tempo))
- Holds an integer, Duration

  Functions:
- Getters and Setters for all variables
- public void print()
- public boolean isContiguous(Note aNote)
- public boolean isEquals(Note aNote)
- public boolean overlaps(Note aNote)
- public boolean isSimultanious(Note aNote)
- public int getDifference(Note aNote, Key theKey)

public class Key (is a singleton)

- Holds a Pitch
- This Pitch is static

Functions:

- Getter and Setter for Pitch

public class Line

- Holds a Note, myStart
- Holds the Key
- Functions as a DoubleLinkedList

public interface CounterPoint

- The interface used by all Species classes

Functions:

- assembleLines() (creates the necessary lines based on the rules specific of the Species)

public class SpeciesOne(through SpeciesFour) implements CounterPoint

- Holds up to three Lines, any unused lines are set to null
- Holds the Key
- Each Species class holds different logic within its functions

Functions:

- public void assemblesLines()
- private void assembleLinesAux()

public class LineInterpreter

Functions:

- public void toText
    - creates a text file with the song as an interpretation
- public void toMIDI
    - creates a MIDI file of the song

We are doing this with the Westergaard version of counterpoint.

"Species counterpoint generally offers less freedom to the composer than other types of counterpoint and therefore is called a 'strict' counterpoint. Species counterpoint was developed as a pedagogical tool in which students progress through several "species" of increasing complexity, with a very simple part that remains constant known as the cantus firmus (Latin for "fixed melody"). The student gradually attains the ability to write free counterpoint (that is, less rigorously constrained counterpoint, usually without a cantus firmus) according to the given rules at the time.[3] The idea is at least as old as 1532, when Giovanni Maria Lanfranco described a similar concept in his Scintille di musica (Brescia, 1533). The 16th-century Venetian theorist Zarlino elaborated on the idea in his influential Le institutioni harmoniche, and it was first presented in a codified form in 1619 by Lodovico Zacconi in his Prattica di musica. Zacconi, unlike later theorists, included a few extra contrapuntal techniques, such as invertible counterpoint.

In 1725 Johann Joseph Fux published Gradus ad Parnassum (Steps to Parnassus), in which he described five species:

1. Note against note;
2. Two notes against one;
3. Four (modified by others to include three, six, etc.) notes against one;
4. Notes offset against each other (as suspensions);"

THIS WAS COPIED DIRECTLY FROM THE WIKIPEDIA ARTICLE ON COUNTERPOINT :
http://en.wikipedia.org/wiki/Counterpoint#Species_counterpoint

Two different kinds of sound handling we may use:
http://kevinboone.net/javamidi.html
http://www.oracle.com/technetwork/java/index-jsp-140234.html