

Figure . Datatype Hierarchy

The datatypes for the project have been designed in an attempt to model actual music. Each piece consists of multiple voices, which consist of measure, repeat, and ending objects. These objects are designed so that playing the piece will only require a call to a single method, which will then make calls to the requisite music object.

The voice object will contain a list of measures, repeats, and endings. Based on the type of repeat object, the voice object will organize measures into an immutable list. Each measure will contain a series of phrases and rests. These phrases will then be decomposed into note objects, such as Tuple, SingleNote, and Chord which can be concatenated into an ordered list. Once this ordered list of notes is created, they will be passed into the music sequencer, and the piece will be played.

Each of the objects that implements the Phrase interface will contain a list of AbstractNotes. These objects are the basis for a note, and contain a single pitch, the length of the pitch, and any accidentals on that pitch. The entire piece is therefore composed implicitly of a list of AbstractNotes. One can use this to construct a list of PlayedNotes using the getNotes() method in AbstractNotes. Finally, one can use the getMidi() method of the PlayedNotes class to get all the midi notes to pass to the sequencer.