Assignment 3

SUMMARY
In this assignment you will develop a class to work with solos in jMusic.

DEADLINE
This assignment is due on Wednesday, December 11 at 11:00 pm.

DESCRIPTION
A LITTLE MUSIC MATH
Music is very mathematical when you break it down. Don't worry if you don't have a music back-
ground. This section should bring you up to speed on what you need to know for the next part.

The key of a song describes which pitches fit into a song, and which do not. A key can be described
by a root and a scale. The root is an integer value pitch that determines the start point of the scale.
The scale is an array of ints that determines which offset pitches are "in-key."

Let's start with the major scale. The int array for the major scale is
{0, 2, 4, 5, 7, 9, 11}.

This is also given by JMC.MAJOR_SCALE.

Now, given any root \(N\), figuring out which pitches are in the key is just a simple math problem. The
following pitches are in the key:
\(N, N+2, N+4, N+5, N+7, N+9, N+11\)

Which means, of course that the following pitches are not in the key:
\(N+1, N+3, N+6, N+8, N+10\)

Take the root of C for example. Using C4 = 60 as an example, the following pitches are in the key of
C Major:
60, 62, 64, 65, 67, 69, 71

Also, C5 = 72, so the following pitches are also in the key of C Major:
72, 74, 76, 77, 79, 81, 83

In fact, adding or subtracting 12 to any root gives the same note letter in a different octave, so we can
use this formula to determine if any pitch \(P\) (0 ≤ P ≤ 127) is in the key or not.

Furthermore, this can be done with any root \(R\) (0 ≤ R ≤ 127), and any scale given as an array of ints
\(S = \{S_1, S_2, ..., S_n\}\) where 0 ≤ \(S_i\) ≤ 11 for 0 ≤ i ≤ n.
Principles of OO Programming

- **Encapsulation**
  - Objects can combine data and operations

- **Inheritance**
  - Classes can inherit properties from other classes

- **Polymorphism**
  - Objects can determine appropriate operations at execution time

Abstract Classes

- **Example**
  - CD player and DVD player
    - Both involve an optical disk
    - Operations
      - Insert, remove, play, record, and stop such discs

Abstract Classes

- **Abstract classes**
  - An abstract class is used only as the basis for subclasses
    - It defines a minimum set of methods and data fields for its subclasses
  - An abstract class has no instances
  - An abstract class should, in general, omit implementations except for the methods that
    - Provide access to private data fields
    - Express functionality common to all of the subclasses

CDP and DVDP have an abstract base class GDP
Abstract Classes

- Abstract classes (continued)
  - A class that contains at least one abstract method must be declared as an abstract class
  - A subclass of an abstract class must be declared abstract if it does not provide implementations for all abstract methods in the superclass

Java Interfaces

- A Java interface
  - Specifies the common behavior of a set of classes
  - Common uses
    - Facilitate moving from one implementation of a class to another
    - A client can reference a class’s interface instead of the class itself
    - Specify behaviors that are common to a group of classes

Java Interfaces

- Inheritance can be used to define a subinterface
- The Java API provides many interfaces and subinterfaces
  - Example: java.util.Iterable
    - An iterator is a class that provides access to another class that contains many objects