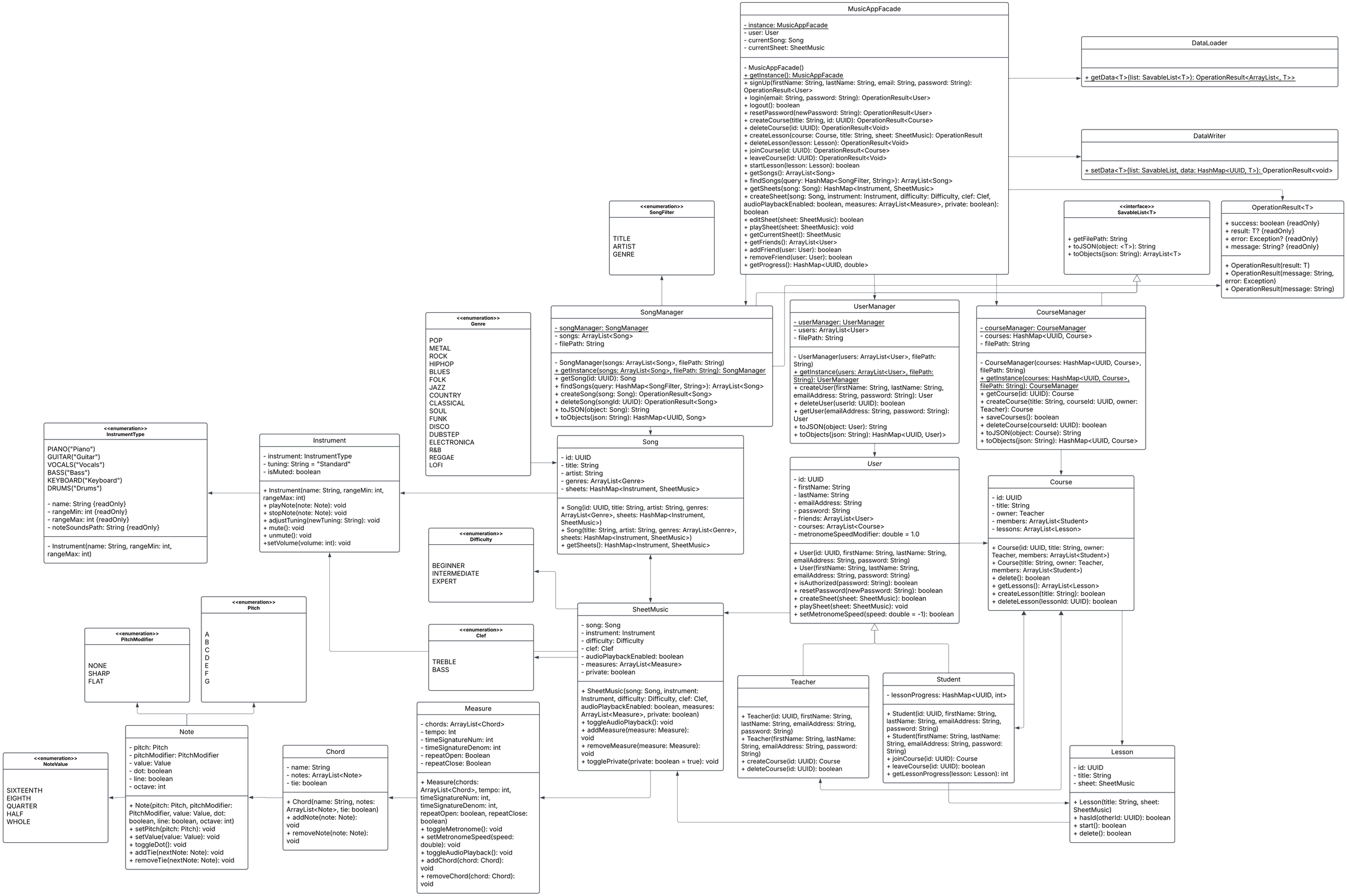
MusicAppFacade
- instance: MusicAppFacade
- user: User
- currentSong: Song
- currentSheet: SheetMusic
- MusicAppFacade()
+ getInstance(): MusicAppFacade
+ signUp(firstName: String, lastName: String, email: String, password: String): OperationResult<User>
+ login(email: String, password: String): OperationResult<User>
+ logout(): boolean
+ resetPassword(newPassword: String): OperationResult<User>
+ createCourse(title: String, id: UUID): OperationResult<Course>
+ deleteCourse(id: UUID): OperationResult<Void>
+ createLesson(course: Course, title: String, sheet: SheetMusic): OperationResult
+ deleteLesson(lesson: Lesson): OperationResult<Void>
+ joinCourse(id: UUID): OperationResult<Course>
+ leaveCourse(id: UUID): OperationResult<Void>
+ startLesson(lesson: Lesson): boolean
+ getSongs(): ArrayList<Song>
+ findSongs(query: HashMap<SongFilter, String>): ArrayList<Song>
+ getSheets(song: Song): HashMap<Instrument, SheetMusic>
+ createSheet(song: Song, instrument: Instrument, difficulty: Difficulty, clef: Clef, audioPlaybackEnabled: boolean, measures: ArrayList<Measure>, private: boolean): boolean
+ editSheet(sheet: SheetMusic): boolean
+ playSheet(sheet: SheetMusic): void
+ getCurrentSheet(): SheetMusic
+ getFriends(): ArrayList<User>
+ addFriend(user: User): boolean
+ removeFriend(user: User): boolean
+ getProgress(): HashMap<UUID, double>
DataLoader
+ getData<T>(list: SavableList<T>): OperationResult<ArrayList<, T>>
DataWriter
+ setData<T>(list: SavableList, data: HashMap<UUID, T>): OperationResult<void>
<<enumeration>>
SongFilter
TITLE
ARTIST
GENRE
<<interface>>
SavableList<T>
+ getFilePath: String
+ toJSON(object: <T>): String
+ toObjects(json: String): ArrayList<T>
OperationResult<T>
+ success: boolean {readOnly}
+ result: T? {readOnly}
+ error: Exception? {readOnly}
+ message: String? {readOnly}
+ OperationResult(result: T)
+ OperationResult(message: String, error: Exception)
+ OperationResult(message: String)
<<enumeration>>
Genre
POP
METAL
ROCK
HIPHOP
BLUES
FOLK
JAZZ
COUNTRY
CLASSICAL
SOUL
FUNK
DISCO
DUBSTEP
ELECTRONICA
R&B
REGGAE
LOFI
SongManager
- songManager: SongManager
- songs: ArrayList<Song>
- filePath: String
- SongManager(songs: ArrayList<Song>, filePath: String)
+ getInstance(songs: ArrayList<Song>, filePath: String): SongManager
+ getSong(id: UUID): Song
+ findSongs(query: HashMap<SongFilter, String>): ArrayList<Song>
+ createSong(song: Song): OperationResult<Song>
+ deleteSong(songId: UUID): OperationResult<Song>
+ toJSON(object: Song): String
+ toObjects(json: String): HashMap<UUID, Song>
UserManager
- userManager: UserManager
- users: ArrayList<User>
- filePath: String
- UserManager(users: ArrayList<User>, filePath: String)
+ getInstance(users: ArrayList<User>, filePath: String): UserManager
+ createUser(firstName: String, lastName: String, emailAddress: String, password: String): User
+ deleteUser(userId: UUID): boolean
+ getUser(emailAddress: String, password: String): User
+ toJSON(object: User): String
+ toObjects(json: String): HashMap<UUID, User)>
CourseManager
- courseManager: CourseManager
- courses: HashMap<UUID, Course>
- filePath: String
- CourseManager(courses: HashMap<UUID, Course>, filePath: String)
+ getInstance(courses: HashMap<UUID, Course>, filePath: String): CourseManager
+ getCourse(id: UUID): Course
+ createCourse(title: String, courseId: UUID, owner: Teacher): Course
+ saveCourses(): boolean
+ deleteCourse(courseId: UUID): boolean
+ toJSON(object: Course): String
+ toObjects(json: String): HashMap<UUID, Course>
<<enumeration>>
InstrumentType
PIANO("Piano")
GUITAR("Guitar")
VOCALS("Vocals")
BASS("Bass")
KEYBOARD("Keyboard")
DRUMS("Drums")
- name: String {readOnly}
- rangeMin: int {readOnly}
- rangeMax: int {readOnly}
- noteSoundsPath: String {readOnly}
- Instrument(name: String, rangeMin: int, rangeMax: int)
Instrument
- instrument: InstrumentType
- tuning: String = "Standard"
- isMuted: boolean
+ Instrument(name: String, rangeMin: int, rangeMax: int)
+ playNote(note: Note): void
+ stopNote(note: Note): void
+ adjustTuning(newTuning: String): void
+ mute(): void
+ unmute(): void
+setVolume(volume: int): void
Song
- id: UUID
- title: String
- artist: String
- genres: ArrayList<Genre>
- sheets: HashMap<Instrument, SheetMusic>
+ Song(id: UUID, title: String, artist: String, genres: ArrayList<Genre>, sheets: HashMap<Instrument, SheetMusic>)
+ Song(title: String, artist: String, genres: ArrayList<Genre>, sheets: HashMap<Instrument, SheetMusic>)
+ getSheets(): HashMap<Instrument, SheetMusic>
User
- id: UUID
- firstName: String
- lastName: String
- emailAddress: String
- password: String
- friends: ArrayList<User>
- courses: ArrayList<Course>
- metronomeSpeedModifier: double = 1.0
+ User(id: UUID, firstName: String, lastName: String, emailAddress: String, password: String)
+ User(firstName: String, lastName: String, emailAddress: String, password: String)
+ isAuthorized(password: String): boolean
+ resetPassword(newPassword: String): boolean
+ createSheet(sheet: SheetMusic): boolean
+ playSheet(sheet: SheetMusic): void
+ setMetronomeSpeed(speed: double = -1): boolean
Course
- id: UUID
- title: String
- owner: Teacher
- members: ArrayList<Student>
- lessons: ArrayList<Lesson>
+ Course(id: UUID, title: String, owner: Teacher, members: ArrayList<Student>)
+ Course(title: String, owner: Teacher, members: ArrayList<Student>)
+ delete(): boolean
+ getLessons(): ArrayList<Lesson>
+ createLesson(title: String): boolean
+ deleteLesson(lessonId: UUID): boolean
<<enumeration>>
Difficulty
BEGINNER
INTERMEDIATE
EXPERT
<<enumeration>>
Pitch
A
B
C
D
E
F
G
<<enumeration>>
PitchModifier
NONE
SHARP
FLAT
<<enumeration>>
Clef
TREBLE
BASS
SheetMusic
- song: Song
- instrument: Instrument
- difficulty: Difficulty
- clef: Clef
- audioPlaybackEnabled: boolean
- measures: ArrayList<Measure>
- private: boolean
+ SheetMusic(song: Song, instrument: Instrument, difficulty: Difficulty, clef: Clef, audioPlaybackEnabled: boolean, measures: ArrayList<Measure>, private: boolean)
+ toggleAudioPlayback(): void
+ addMeasure(measure: Measure): void
+ removeMeasure(measure: Measure): void
+ togglePrivate(private: boolean = true): void
Teacher
+ Teacher(id: UUID, firstName: String, lastName: String, emailAddress: String, password: String)
+ Teacher(firstName: String, lastName: String, emailAddress: String, password: String)
+ createCourse(id: UUID): Course
+ deleteCourse(id: UUID): boolean
Student
- lessonProgress: HashMap<UUID, int>
+ Student(id: UUID, firstName: String, lastName: String, emailAddress: String, password: String)
+ Student(firstName: String, lastName: String, emailAddress: String, password: String)
+ joinCourse(id: UUID): Course
+ leaveCourse(id: UUID): boolean
+ getLessonProgress(lesson: Lesson): int
<<enumeration>>
NoteValue
SIXTEENTH
EIGHTH
QUARTER
HALF
WHOLE
Note
- pitch: Pitch
- pitchModifier: PitchModifier
- value: Value
- dot: boolean
- line: boolean
- octave: int
+ Note(pitch: Pitch, pitchModifier: PitchModifier, value: Value, dot: boolean, line: boolean, octave: int)
+ setPitch(pitch: Pitch): void
+ setValue(value: Value): void
+ toggleDot(): void
+ addTie(nextNote: Note): void
+ removeTie(nextNote: Note): void
Chord
- name: String
- notes: ArrayList<Note>
- tie: boolean
+ Chord(name: String, notes: ArrayList<Note>, tie: boolean)
+ addNote(note: Note): void
+ removeNote(note: Note): void
Measure
- chords: ArrayList<Chord>
- tempo: Int
- timeSignatureNum: int
- timeSignatureDenom: int
- repeatOpen: Boolean
- repeatClose: Boolean
+ Measure(chords: ArrayList<Chord>, tempo: int, timeSignatureNum: int, timeSignatureDenom: int, repeatOpen: boolean, repeatClose: boolean)
+ toggleMetronome(): void
+ setMetronomeSpeed(speed: double): void
+ toggleAudioPlayback(): void
+ addChord(chord: Chord): void
+ removeChord(chord: Chord): void
Lesson
- id: UUID
- title: String
- sheet: SheetMusic
+ Lesson(title: String, sheet: SheetMusic)
+ hasId(otherId: UUID): boolean
+ start(): boolean
+ delete(): boolean