DESCRIPCIÓN

Se quiere crear una aplicación para crear y modificar partituras musicales. Las partituras serán únicamente de una hoja, por lo que es aconsejable un mínimo de 5 pentagramas y es obligatorio un máximo de 20 (podrían modificarse estos números en el futuro). Una partitura se distinguirá de otra según su nombre y sus pentagramas. Además, la clase Partitura será abstracta y tendrá dos clases hijas, ConAutor o Anonima, con un atributo diferente cada una.

Si el usuario desea ver una partitura primero se le mostrarán los datos de forma simple(toString) y luego se le preguntará si quiere la parte musical en detalle (nombre de partitura y pentagramas con sus notas).

- La partitura de + nombre + tiene x pentagramas.
- nombre: Pentagrama 1 + notas(nota, nota, nota…)
  Pentagrama 2 + notas
  …

Los pentagramas deben tener un compás(2, 3 ó 4), un conjunto de notas (cuyo número vendrá del compás) y si está silenciado o no, si lo está se indicará al mostrar la partitura. Dos pentagramas serán iguales si tienen el mismo compás y las mismas notas (será común que haya dos iguales, una partitura puede tener varios pentagramas iguales). Podrá modificarse un pentagrama entero (modificarPentagrama) o una de las notas (modificarNota).

Las notas tienen un nombre que sólo puede ser: do, re mi, fa, sol, la o si, que podrían necesitar cambiarse a la notación inglesa. Además tienen una altura baja o alta (en caso de ser alta se mostrará como true y se pondrá un “ ‘ “ detrás del nombre de la nota), y un tipo, que recoge la duración. Dependiendo de la duración tendremos una nota blanca (duración 2) o una nota negra (duración 1), que se tendrá en cuenta a la hora de añadir notas a un pentagrama, pues la suma de la duración no puede ser mayor al número de compás:

- Compás 2: 2 negras o 1 blanca
- Compás 3: 1 blanca y 1 negra o 3 negras
- Compás 4: 2 blancas, 1 blanca y 2 negras o 4 negras

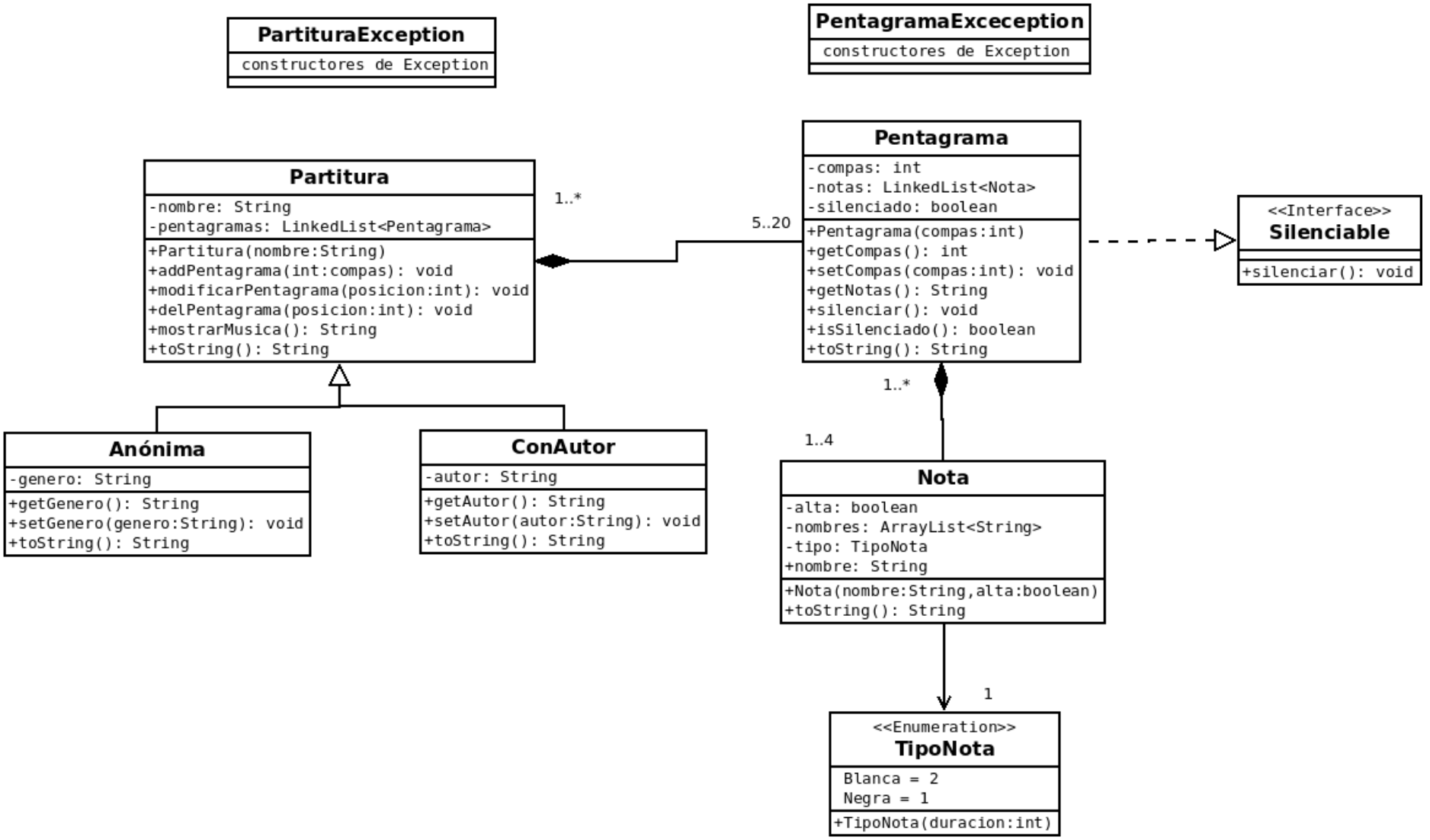
Dos notas serán iguales si tienen el mismo nombre, altura y tipo. Para mostrar una nota se usará el siguiente formato:

- si es alta: nombre + ‘ + tipo sol‘ blanca
- si no es alta: nombre + tipo mi negra

Se deberán manejar los errores con Exception e implementar los métodos HashCode y equals. Se probarán las clases y métodos en el Principal.
Se usará LinkedList porque se borrarán pentagramas y notas en cualquier posición (siendo muy probable es que se modifiquen los de en medio).

*A la hora de empezar a programar me he dado cuenta de que tenía que cambiar o añadir algunos métodos. ¿Cuando acabe el proyecto tengo que subir la versión actualizada del uml?

UML


PartituraException
constructores de Exception
PentagramaExceception
constructores de Exception
Partitura
-nombre: String
-pentagramas: LinkedList<Pentagrama>
+Partitura(nombre:String)
+addPentagrama(int:compas): void
+modificarPentagrama(posicion:int): void
+delPentagrama(posicion:int): void
+mostrarMusica(): String
+toString(): String
1..*
5..20
Pentagrama
-compas: int
-notas: LinkedList<Nota>
-silenciado: boolean
+Pentagrama(compas:int)
+getCompas(): int
+setCompas(compas:int): void
+getNotas(): String
+silenciar(): void
+isSilenciado(): boolean
+toString(): String
<<Interface>>
Silenciable
+silenciar(): void
Anónima
-genero: String
+getGenero(): String
+setGenero(genero:String): void
+toString(): String
ConAutor
-autor: String
+getAutor(): String
+setAutor(autor:String): void
+toString(): String
1..*
1..4
Nota
-alta: boolean
-nombres: ArrayList<String>
-tipo: TipoNota
+nombre: String
+Nota(nombre:String,alta:boolean)
+toString(): String
1
<<Enumeration>>
TipoNota
Blanca = 2
Negra = 1
+TipoNota(duracion:int)