Exam of Programmazione II, February 20, 2025

(please submit `AbstractNote.java`, `AbstractSong.java`, `BasicSong.java`, `EnglishNote.java`, `IllegalNoteException.java`, `ItalianNote.java` and `Scale.java`)

Create en Eclipse project and the `it.univr.notes` package. Copy the classes of the exam inside that package. You cannot change the declaration of methods and classes. You can define extra fields, methods or constructors not required in the assignment, as long as they are `private`. You can define extra classes, that must be submitted at the end. The submitted solution must compile, or it will be discarded.

The interface `Note`, already completed, represents a musical note by its semitone: low notes have a smaller semitone and more acute notes have a higher semitone. Italian and English notes are different for the way they are represented by their `toString`, as the following table shows:

`ItalianNote`	semitone	`EnglishNote`
1.do	0	1.C
1.do#	1	1.C#
1.re	2	1.D
1.re#	3	1.D#
1.mi	4	1.E
1.fa	5	1.F
1.fa#	6	1.F#
1.sol	7	1.G
1.sol#	8	1.G#
1.la	9	1.A
1.la#	10	1.A#
1.si	11	1.B
2.do	12	2.C
2.do#	13	2.C#
2.re	14	2.D
2.re#	15	2.D#
2.mi	16	2.E
2.fa	17	2.F
2.fa#	18	2.F#
2.sol	19	2.G
2.sol#	20	2.G#
2.la	21	2.A
2.la#	22	2.A#
2.si	23	2.B
3.do	24	3.C
3.do#	25	3.C#
eccetera	eccetera	eccetera

Notes have a `shift` method that allows one to obtain a note from another note, by increasing or by decreasing its semitone. For instance, (1.la).`shift`(3) yields 2.do, while (1.G).`shift`(8) yields 2.D# and (2.do).`shift`(-1) yields 1.si.

Exercise 1 (2 points). Define the checked `IllegalNoteException`, used to mark that the user is trying to create a note outside the bounds 0 . . . `Note.MAX_SEMITONE`.

Exercise 2 (3 points). Complete class `AbstractNote.java`, that contains the code shared among the concrete implementations of the notes.

Exercise 3 (5 points). Complete the subclasses `ItalianNote.java` and `EnglishNote.java`, that are different for their methods `toString` and `shift`. The latter yields an `ItalianNote` when applied to an `ItalianNote` and yields an `EnglishNote` when applied to an `EnglishNote`.

Exercise 4 (5 points). The `Song` interface (already completed, do not modify) represents a song, that is, a sequence of notes. By iterating on a `Song`, one gets its notes. Complete the `AbstractSong.java` class with the code shared among the concrete implementations of a song, that is, its `toString` method, that appends the `toString` of the notes of the song, with a space as separator.

Exercise 5 (7 points). Complete the `BasicSong` class, that implements a song whose notes are explicitly provided to its constructor and stored inside a list of notes.

Exercise 6 (9 points). Complete the `Scale` class, that implements a song whose notes are obtained as a scale that starts from a note `start`, provided to the constructor, and continues upwards, semitone by semitone, for a total of 12 notes. It is required to solve this exercise without adding fields to the class `Scale`, beyond those that you will find already defined there.

To understand better, consider the following `Main.java`, already written, not to modify:

```
public class Main {

  public static void main(String[] args) throws IllegalNoteException {
    // a song made of 4 notes
    Song s1 = new BasicSong
      (new ItalianNote(5), new ItalianNote(25), new EnglishNote(13), new EnglishNote(38));
    // a scale from the English note of semitone 20 upwards, until the English note with semitone 31
    Song s2 = new Scale(new EnglishNote(20));
    System.out.println("s1 = " + s1); // print s1
    System.out.println("s2 = " + s2); // print s2
    System.out.println("s1.shift(-3) = " + s1.shift(-3)); // print s1 decreased by three semitones
    System.out.println("s2.shift(4) = " + s2.shift(4)); // print s2 increased by four semitones
    System.out.println("s1 = " + s1); // print s1 again (it won't be changed)
    System.out.println("s2 = " + s2); // print s2 again (it won't be changed)
    s2.shift(29); // exception: by shifting s2's last note one goes outside the bounds 0...MAX_SEMITON
  }
}
```

Its execution should print:

```
1 = 1.fa 3.do# 2.C# 4.D
s2 = 2.G# 2.A 2.A# 2.B 3.C 3.C# 3.D 3.D# 3.E 3.F 3.F# 3.G
s1.shift(-3) = 1.re 2.la# 1.A# 3.B
s2.shift(4) = 3.C 3.C# 3.D 3.D# 3.E 3.F 3.F# 3.G 3.G# 3.A 3.A# 3.B
s1 = 1.fa 3.do# 2.C# 4.D
s2 = 2.G# 2.A 2.A# 2.B 3.C 3.C# 3.D 3.D# 3.E 3.F 3.F# 3.G
Exception in thread "main" it.univr.notes.IllegalNoteException
```