

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

PROJECT

Object-Oriented Programming

Electronic Piano

|  |  |
| --- | --- |
| ***Instructor:*** | ***Prof.Nguyen Thi Thu Trang*** |

*Group 12: Ngo Thi Thu Huyen 20200289*

*Vu Tan Khai 20200311*

*Nguyen Duy Khanh 20204914*

*Nguyen Ngoc Khanh 20204915*

I.Assignment of members:

II.Mini-project description:

1.Mini-project requirement:

-Implement an application that provides GUI for the user to virtually play an electronic piano.

-Design requirement:

+ On the main menu: title of the application, piano GUI, help menu, quit

• User can play the piano by interacting with GUI

• Help menu shows the basic usage and aim of the program

• Quit exits the program. Ask for confirmation to quit

+Demonstration:

• Keyboard: C (Do), D (Re), E (Mi), F (Fa), G (Sol), A (La), B (Xi). The keyboard is designed to be as much details as possible.

• A button for increasing/decreasing volume (optional)

• A record button to record the play (optional)

• A button for changing music style (optional)

2.Use case diagram:

Diagram

Description automatically generated

Explanation:

-Play piano by keyboard: Users are able to play the electronic piano by pressing keyboard button. Which keyboard button corresponds to which piano key is shown on the piano.

-Play piano by mouse: User can also play the electronic piano by left clicking on the piano key.

-Adjust volume: there is a volume slider for user to increase or decrease the volume of the sound.

-View help menu: user can click on the help button to get the instructions of using the app.

-Record the play: allows user to record their play and then output a .wav file.

-Replay the record: user can replay their record by clicking on play button after clicking on record button.

-Change music style: there are 3 styles of sound for user to choose (piano, guitar, organ).

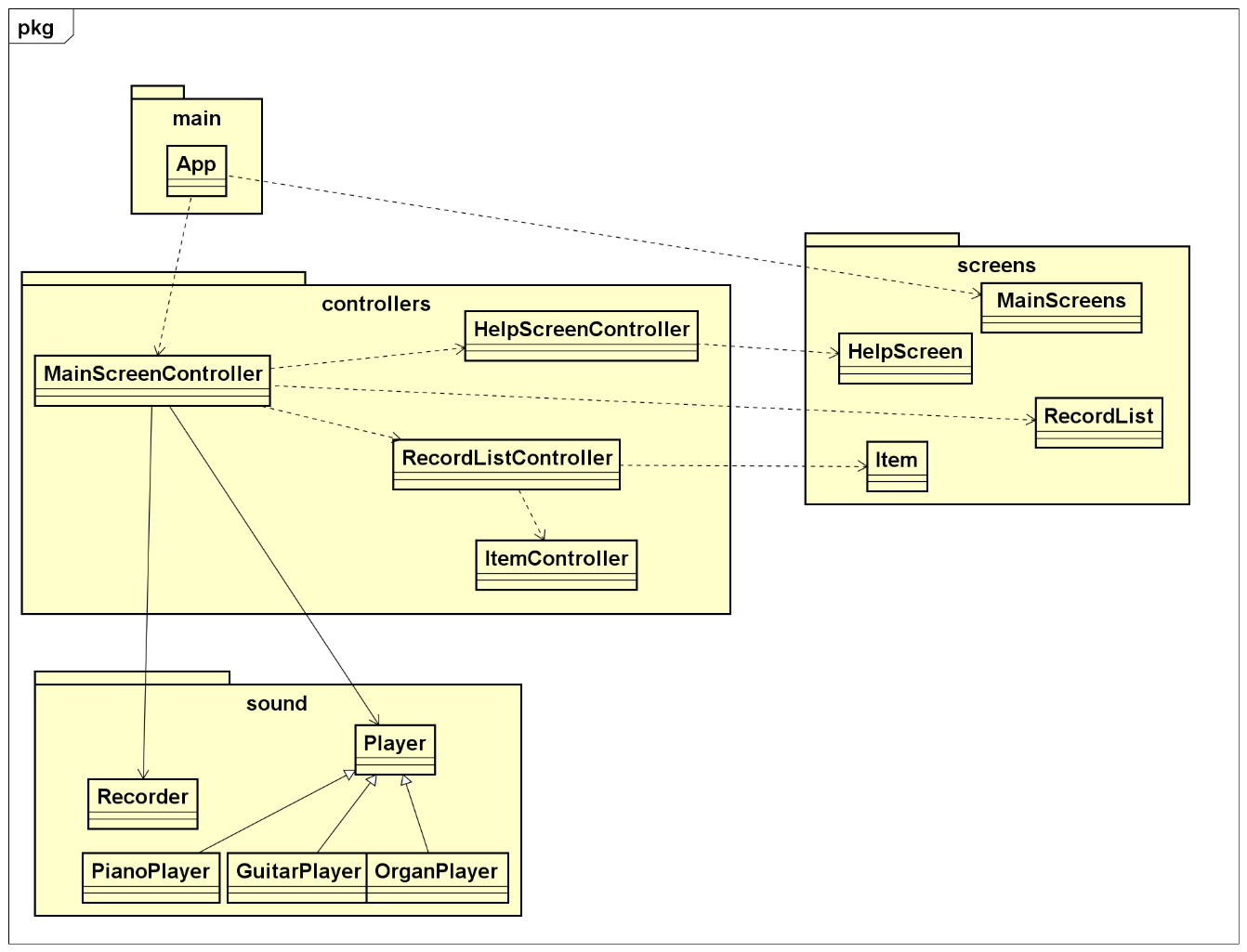
-Create music sheet: Each music key user play is logged in the music sheet (the white area right above the piano keyboard). Click on Clear button to restart a new sheet.

-Replay music sheet: click on Replay button to automatically play the music sheet.

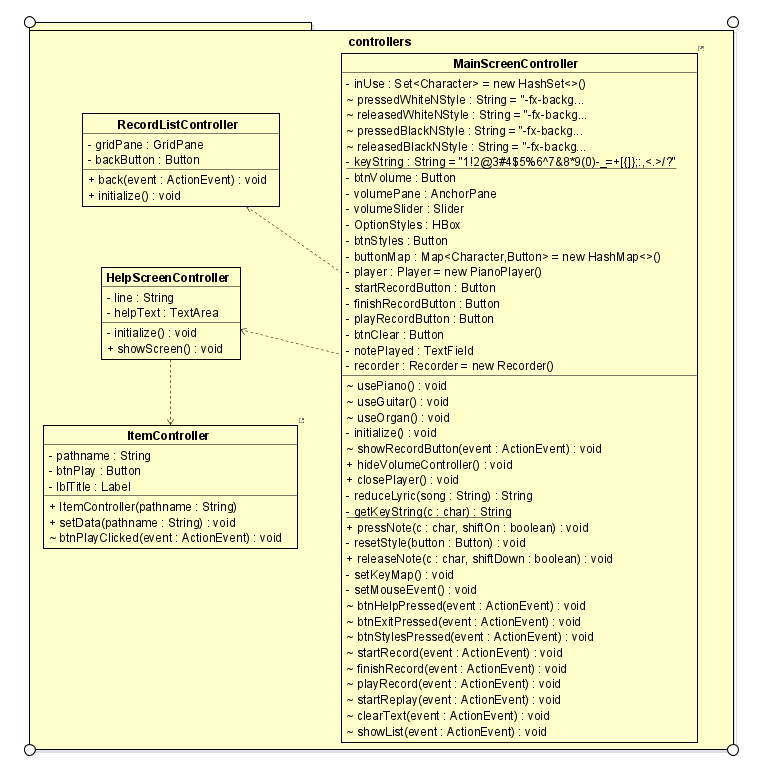
-Quit the application: click on exit button to terminate the program. User is asked for comfimation.

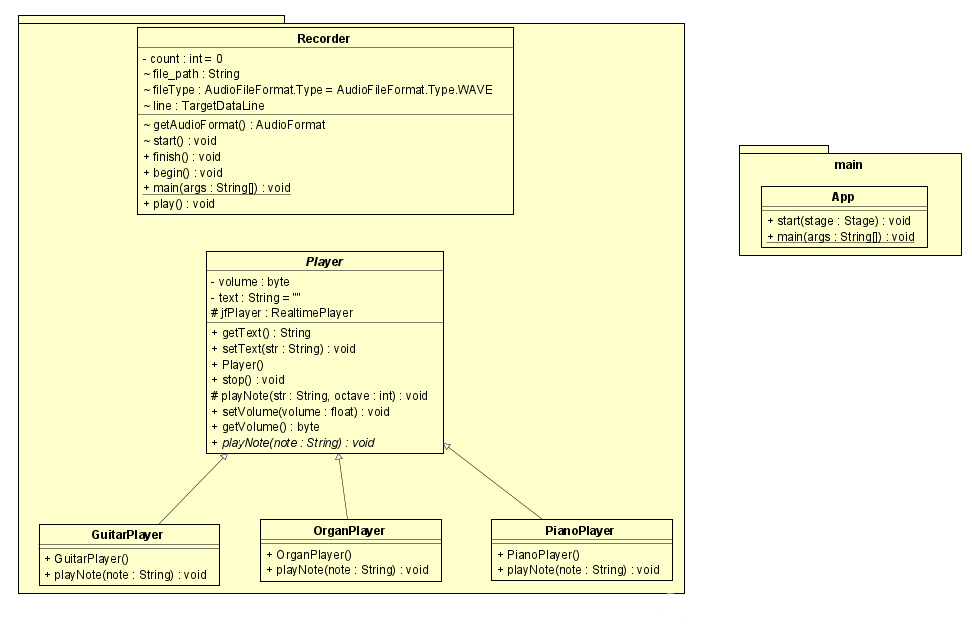
III.Design:

1.General class diagram:



2.Detail class diagram:



 3.Explanation of the design:

-Inheritance: Our application allows user choosing how piano sounds from three types: guitar, organ and piano. Then, we create three classes corresponding to each type of the sound respectively: GuitarPlayer, OrganPlayer and PianoPlayer. These three classes all share same methods of playing note(playNote(note:String), playNote(str:String, octave:int)), adjusting the volume of the sound (setVolume(volume:float), getVolume(volume: float)), stop playing(stop()) and create the music sheet(getText(), setText(str:String)). To promote the reusability of the code, from that also enhance its reliability, we create an abstract class Player containing all shared methods and let GuitarPlayer, OrganPlayer, PianoPlayer inherit. To force each subclass to override playNote(note:String): void method, we make this method to be abstract method , that’s why we let Player be an abstract class.

-Polymorphism:

Text

Description automatically generated

Text

Description automatically generated with medium confidence

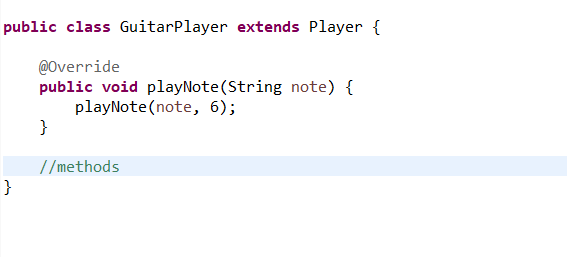
+Method polymorphism(method overloading): each instance of the class Player has two playNote methods with different signature: playNote(str: String, octave: int), playNote(note: String). The main difference between two methods is that while playNote(str: String, octave: int) let the programmer decide the octave parameter which is pitch[[1]](#footnote-1) of the sound, the other fixes the octave parameter. Instead of having two methods with different name by changing playNote(str: String, octave: int) method to playNoteOctaveFive or playNoteOctaveFour or something similar to it, we take the advantage of method overloading technique to have two methods with the same name. So that , we successfully enhance the readability of the code.

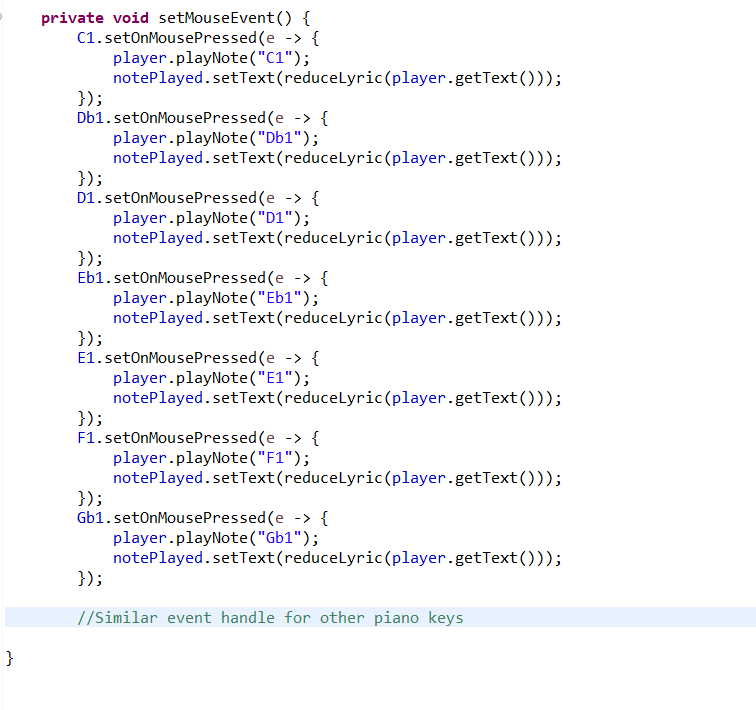
+Object polymorphism:

Text

Description automatically generated

Text

Description automatically generated



In the source code above, we can easily see the polymorphism in the player.playNote() statement. Thanks to the help of dynamic binding at runtime, instead of using three variables to store different types of player : PianoPlayer, OrganPlayer, GuitarPlayer, we can use a single player variable of Player type. So that, polymorphism reduces the size of the code, make it less complicated. As the result, it is easier to debug the code in the future.

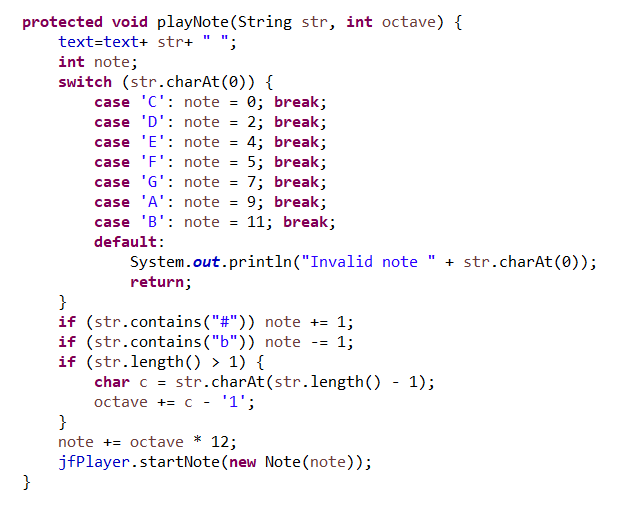
-Association: Notice that in our implementation, MainScreenController class has two fields (player: Player and recoder: Recorder) so that it models the association between MainScreenController and Player, MainScreenController and Recorder. MainScreenController use Player object to control of how the application sound and log the music sheet, it also use Recorder object to manage all record operations.

4.Implementation of some important methods:

- playNote

Input: str: String indicates a note

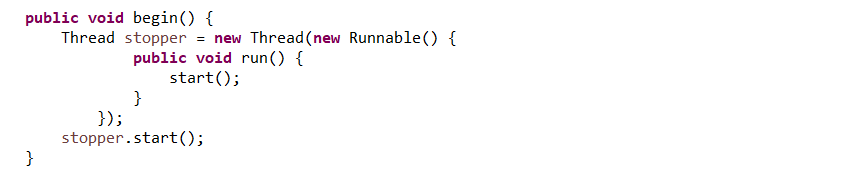
octave: int indicates the pitch of the sound

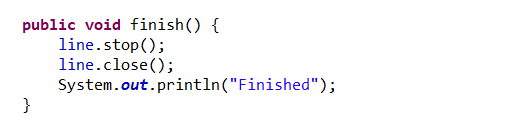


This method is based on JFugue package to play sound. First it append a new note string to text variable (text contains the content of the music sheet). Then the method convert the note string to the corresponding integer number defining the frequency of the sound. If the note string contains “#” character, increment the note variable, if the note string contains “b” character, decrement the note variable. In music theory, “#” indicates a sharp note, “b” indicates a flat note. When a note's pitch is sharpened, it is raised by a semitone (or a half-step). Similarly, when a note's pitch is flattened, it is lowered by a semitone. The method also increase the note variable by octave \* 12 given that each octave is different by 12 units. Finally, it executes jfPlayer.startNote(new Note(note)) statement to play sound.

-Recorder class:







Graphical user interface, text, application, email

Description automatically generated

The Recorder class is dedicated to storing the input sound from the environment into a wav file. The wav is stored in the location stored in the file\_path variable in start method. The whole record procedure is executed in a different thread from the main thread which will increase the performance of the application. The play() method is implemented to help the user play the record they want.

1. In music theory, pitch defines the highness of the sound [↑](#footnote-ref-1)