Piano Application
Java Programming

By
Guruprasath M R(CH.EN.U4AIE22015)
Sudeesh Kumar V(CH.EN.U4AIE22059)

Table of contents

Introduction

- Purpose: Create a piano application with a graphical user interface (GUI) in Java.
- Libraries: Java, Swing, midi.

GUI Components

- Swing: Java's GUI toolkit for creating graphical interfaces.
- JFrame: Main window of the application, providing the container for GUI components.
- JPanel: Container for piano keys, allowing for easy organization and layout management.
- GridLayout: A layout manager used to arrange the piano keys in a grid format.

PianoButton Class

- Extends JButton to represent individual piano keys in the GUI.
- Properties:
  - note: MIDI note value of the key.
  - isBlack: Indicates whether the key is a black key or a white key.
- Methods:
  - playSound(): Plays the note associated with the key.
  - stopSound(): Stops playing the note.
- MIDI Sound Generation:
  - Uses the Synthesizer and MidiChannel classes for sound synthesis.
  - Plays and stops notes based on user interaction.

PianoApp Class

- Responsible for initializing and managing the piano application.
- Contains the main method to start the application.
- GUI Components:
  - JFrame: Main window of the application.
  - JPanel: Container for piano keys.
- KeyListener Implementation:
  - Implements the KeyListener interface to handle keyboard input.
  - Methods: keyPressed(), keyReleased(), keyTyped().
- MIDI Sound Generation:
  - Uses the Synthesizer and MidiChannel classes for sound synthesis.
  - Plays and stops notes based on keyboard input.

KeyListener Implementation

- Implements the KeyListener interface to handle keyboard input for the piano keys.

- Methods:
  - keyPressed(): Triggered when a key is pressed, allowing for real-time sound generation.
  - keyReleased(): Triggered when a key is released, stopping the sound playback.

KeyListener Implementation

- Implements the KeyListener interface to handle keyboard input for the piano keys.

- Methods:
  - keyPressed(): Triggered when a key is pressed, allowing for real-time sound generation.
  - keyReleased(): Triggered when a key is released, stopping the sound playback.

Key Mapping

- Utilizes a HashMap to map keyboard keys to piano keys, enabling accurate note generation.
- Supports lowercase and uppercase keys, allowing for a wider range of inputs.
- Key map for first octave and second octave is upper of given key
- {‘keyboard key’ -> ‘Note’}

  {z -> C, x -> D, c -> E, v -> F, b -> G ,
  n -> A, m -> B, a -> C#,s -> D#,j -> F#,
  k ->G#,l -> A#}
- Supports 2 octaves and variations

MIDI Sound Generation

- Utilizes the `Synthesizer` and `MidiChannel` classes for MIDI sound synthesis.
- Opens the synthesizer and obtains the `MidiChannel` for generating sounds.
- Plays notes using the `noteOn` and `noteOff` commands.
- Velocity Parameter:
-   - Controls the loudness of the note, enabling dynamic expression.
-   - Can be adjusted to achieve different levels of intensity in the sound.
- Sound Duration:
-   - Implements a `playSound()` method with adjustable duration for generating custom note lengths.
-   - Allows for realistic note durations and musical expression.
-   - Provides flexibility to control the length of each played note.

DEMO

CODE LINK

{

END

Thank you

}