Javadoc Lettore Musicale

Pavan, Sow, Zamuner

**Canzone:**

è la classe di base del programma. Contiene solamente costruttori, getters, setters e toString.

Attributi:

File file -> percorso

String title -> titolo della canzone

File img -> percorso dell’img corrispondente

**Raccolta:**

contenitore di canzoni.

Attributi:

ArrayList<Canzone> canzoni -> tutte le canzoni

DefaultListModel<Canzone> modello -> canzoni da mostrare nella *JList*

Metodi:

public void aggiungiCanzone(Canzone c);

public Canzone getSong(int idx);

public DefaultListModel<Canzone> getModello();

ritorna il modello da inserire nella *JList* nel *LettoreView*

public void stampaCanzoni();

stampa a console delle canzoni

**SongPlayer:**

gestisce la canzone che sta suonando. La fa partire, fermare, resettare, stoppare.

Attributi:

Canzone song -> canzone selezionata

Clip clip -> attributo necessario per far partire l’audio

AudioInputStream audioStream -> attributo necessario per far partire l’audio

String status -> in che stato è la canzone (“stop”, “play”, “pause”);

Metodi:

public void changeSong(Canzone song);

richiama il metodo stop(). Cambia la canzone e richiama il metodo play()

public void play();

fa partire la canzone selezionata e setta lo status a “play”

public void stop();

ferma la canzone e setta la clip a null, fondamentalmente eliminandola. Setta lo status a “stop”

public void pause();

ferma la canzone in un punto specifico. Setta lo status a “pause”. Si usa in opposizione al resume()

public void resume();

fa ripartire la canzone dal punto in cui è stata messa in pausa. Setta lo status a “play”. Si usa in opposizione al pause()

public void restart();

fa ripartire la canzone dall’inizio. Setta lo status a “play”

private int getTotalTime();

ritorna la lunghezza della canzone in secondi

public int getElapsedTimePercentage();

ritorna un numero [0, 100] che corrisponde alla percentuale di quanto tempo è passato dall’inizio della canzone.

**LettoreView:**

interfaccia grafica del lettore. Utilizza il Layout null (absolute). Contiene getters

metodi:

public void updateBarra(int val);

aggiorna la barra di durata della canzone ricevendo come parametro la percentuale di canzone già suonata

**LettoreController:**

gestisce gli input dell’utente attraverso l’interfaccia grafica e gestisce di conseguenza il *SongPlayer*.

Attributi:

LettoreView v

Raccolta r

SongPlayer sp

Metodi:

private void registraEvento();

aggiunge gli actionListener

public void actionPerformed(ActionEvent e);

metodo ereditato dall’interfaccia *ActionListener*. Viene chiamato quando viene cliccato un oggetto con un action listener. In base a cosa succede viene gestito il *SongPlayer*:

-click del btnPlay:

-se ho selezionato play e sto già suonando la canzone: niente

-se ho selezionato una canzone diversa da quella in corso:

stop + reset barra -> changeSong -> play

-se non ci sono canzoni selezionate: play della prima (oppure setto di

default la prima canzone nel sp)

-se era in pause: cambio canzone oppure sp.resume() + resume della barra

-click del btnPause:

-se non sta suonando una canzone: niente

-altrimenti: sp.pause() + pause della barra

-click del btnStop:

-se non sta suonando una canzone: niente

-altrimenti: stop + reset barra, rimozione canzone dal *SongPlayer*

public void updateBarra();

ciclo infinito che ogni 1000ms controlla se il *SongPlayer* sia in “play” e nel caso aggiorna la barra progressiva del tempo della canzone richiamando il metodo opportuno del *SongPlayer* e del *LettoreView*. Non è propriamente ideale farlo andare con un ciclo infinito però non è stata trovata altra soluzione