# nerDJ True Star

**Gerador de Música**

**Universidade Federal do Rio Grande do Sul**
**Instituto de Informática**

Técnicas de Construção de Programas
Professor Marcelo Pimenta
Trabalho Prático - FASE 3

Grupo: Nicolle Pimentel Favero

1. Lista de Requisitos Funcionais
    - Abrir interface gráfica.
    - Receber texto principal.
    - Receber parâmetros extras: volume inicial, bpm inicial, instrumento inicial e oitava inicial.
    - Validar parâmetros.
    - Traduzir input para a notação da biblioteca JFugue.
    - Tocar a música correspondente aos caracteres do input.

2. Definições de Classes

    Main

    A classe Main apenas é responsável por carregar a interface no método main.

    Controller

    A classe Controller será responsável por chamar e repassar ao Parser os valores recebidos pela interface gráfica; e por repassar ao MusicHandler o input já traduzido para a linguagem do JFugue para a executar a música e gerar o arquivo midi. Ainda na classe Controller é feito um teste para conferir se a string digitada na interface não é a mesma digitada na vez anterior, para economizar tempo na hora de fazer o parser.

    Atributos:
    - *musicString*: *TextArea*
    - *initialVolume*: *TextField*
    - *initialBPM*: *TextField*
    - *initialOctave*: *TextField*
    - *initialInstrument*: *TextField*
    - *musicHandler*: *MusicHandler*
    - *parser*: *Parser*
    - *previousMusicString*: *String*

    Métodos:
    - *run*(): *void – corresponde ao método que irá rodar quando o usuário clicar no botão "Play Song!". Chama o musicHandler e executa a música.*
    - *compareString*(): *boolean – avalia se a string atual é igual a string anterior antes de fazer o parse.*
    - *createMidi*(): *void – corresponde ao método que irá rodar quando o usuário clicar no botão Create MIDI File. Chama o musicHandler e cria o arquivo.*
    - *getInitialVolume*(): *void – valida o TextField do volume.*
    - *getInitialInstrument*(): *void – valida o TextField do instrumento.*
    - *getInitialOctave*(): *void – valida o TextField da oitava.*
    - *getInitialBPM*(): *void – valida o TextField do BPM.*

WindowHandler

A classe WindowHandler é responsável pelas definições da interface. Ela possui um método para configurar a interface e outro para carregá-la.

Métodos:
- *start(Stage primaryStage): void*
- *load(): void*

Parser

A classe Parser faz o mapeamento da string recebida pela interface para strings que as classes do JFugue possam interpretar para tocar música.

Métodos:
- *parseText(String s, integer volumePadrao, integer bpm): Pattern*
- *setPatternVolume(): void - configura o volume da música.*
- *setPatternInstrument(): void - configura o instrumento da música.*

MusicHandler

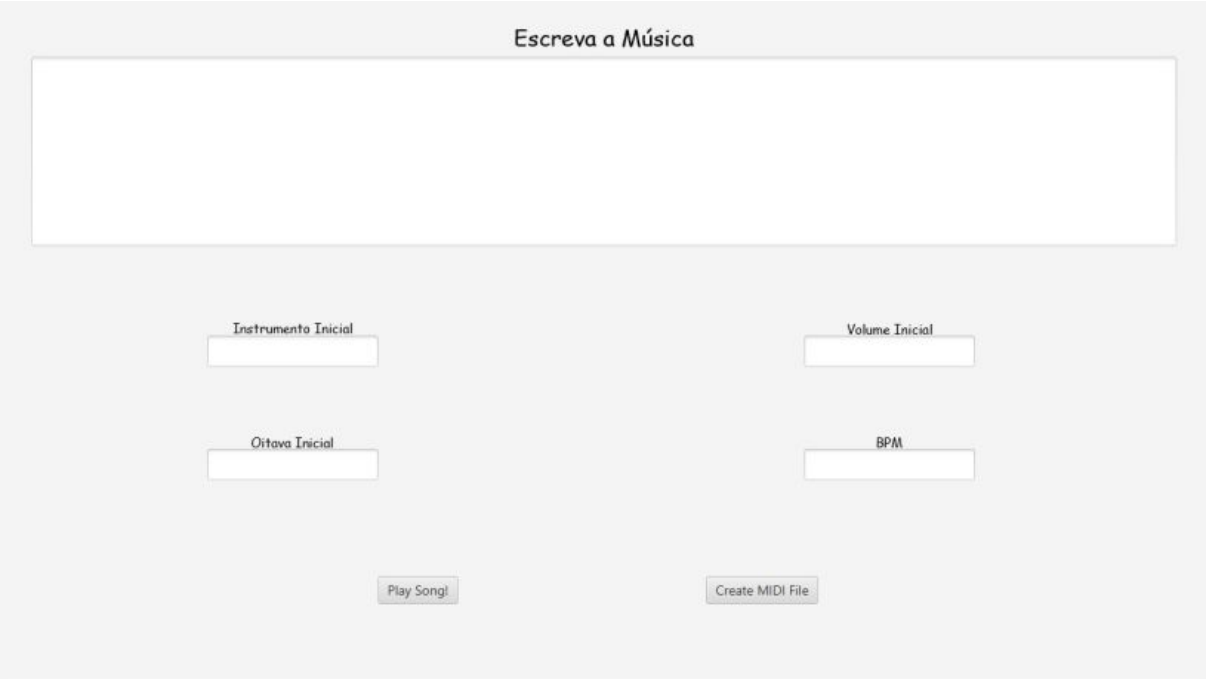
A classe MusicHandler é responsável por, finalmente, tocar a música na string resultante do parse. Além disso, o MusicHandler também cria o arquivo MIDI.

Atributos:
- *player: Player*
- *pattern: Pattern*

Métodos:
- *playSong(): void*
- *saveSong(): void*
- *setPattern(): void*

3. Interface com o Usuário

Foi optado uma interface simples, com JavaFX, onde fosse possível, principalmente, escrever a music string e, de modo auxiliar, valores iniciais de volume, instrumento, oitava e bpm. O nome do arquivo MIDI é previamente definido pelo programa como “song.midi”.




4. Código fonte
   O código fonte do projeto está anexado ao moodle em um arquivo .txt.

5. Testes
   O programa foi sendo testado conforme foi sendo escrito. Foram utilizadas algumas strings de teste (feitas pelo grupo mesmo) como mostrado abaixo para testar as principais funcionalidades do programa.

   **"Aa BbCcccccDdEeFfGg\nBb?CcDdEeFfG..9A>A<;BBBB"**

6. Fonte da capa
   https://wallpapercave.com/true-damage-yasuo-wallpapers