Conversor Midi → JavaClass

Introdução à Computação Sônica - Trabalho 3

Responsáveis: Hélio Santana - 140142959
Rodrigo Guimarães - 140170740

Sumário

- Problema;
- Diagrama de Classes;
- Aquisição Arquivo *Midi*;
- Conversão para *JavaClass*;
- Interface Gráfica;
- Repertório *Online*;
- Referências;
- Executando.

Problema

- Escreva um programa que: Converta arquivos *Midi* em programas *Java* equivalentes, os quais deverão fazer uso de classes do pacote sintese. Em sequida, cada programa *Java* assim gerado deverá ser compilado normalmente (ao estilo do *Trabalho 2*), mas com a diferença de que, no caso deste trabalho, o código será produzido de maneira automatizada.

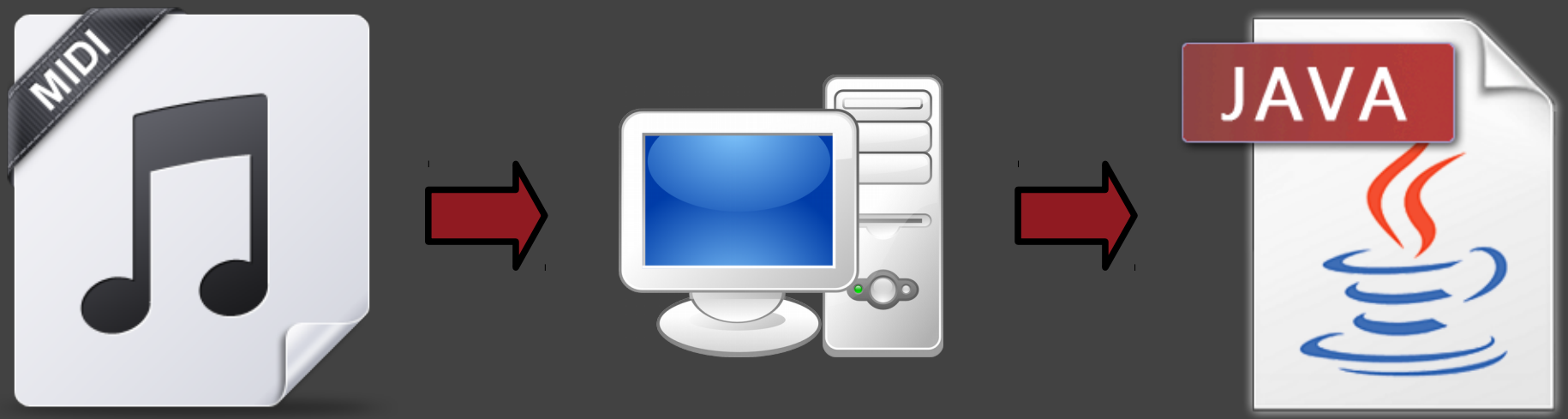
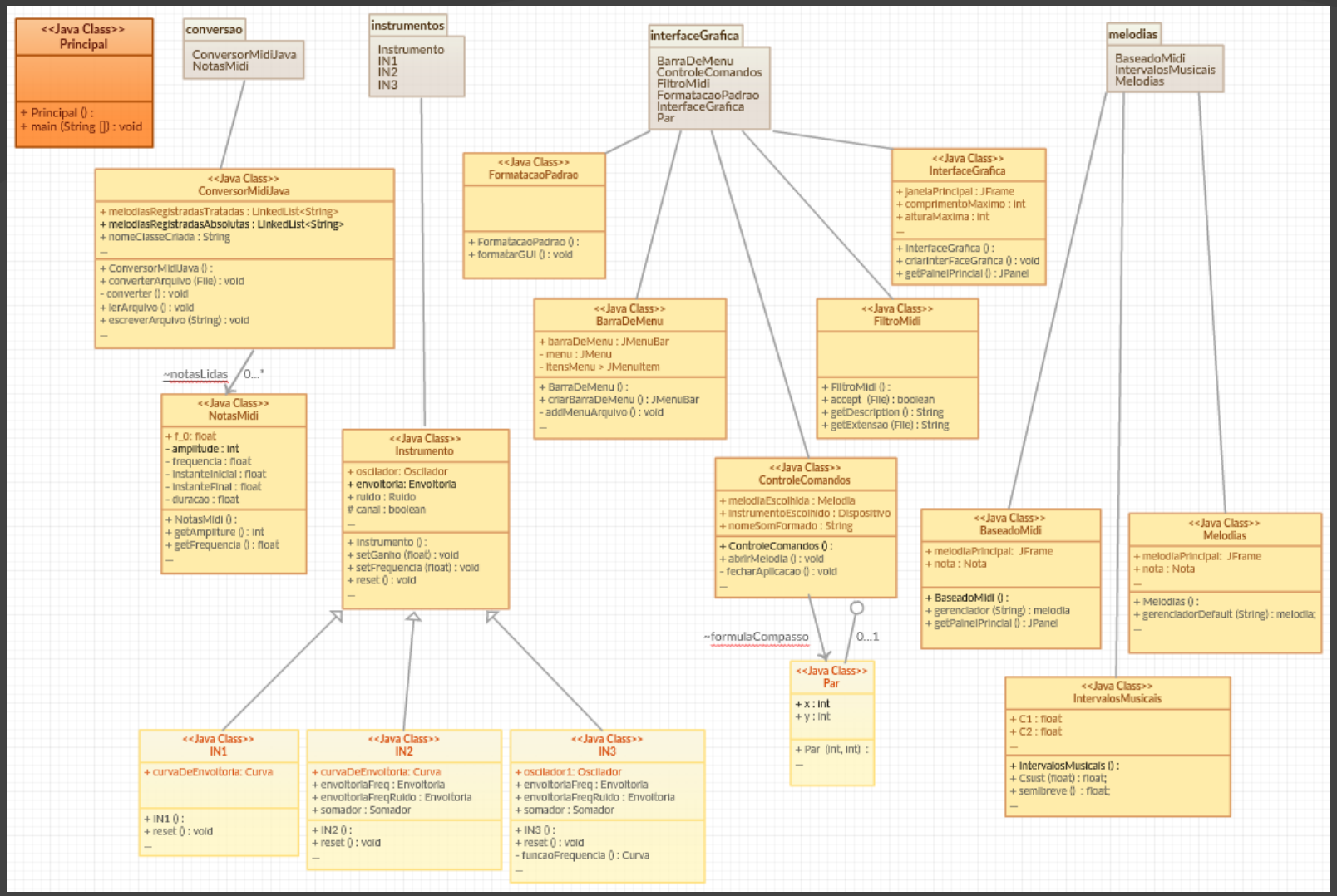


Diagrama de Classes



Conversor Midi → JavaClass

Aquisição Arquivo *Midi*

- Foi considerado a melodia do arquivo, onde uma nota foi caracterizada como tendo a tríplice comum: frequência, amplitude e duração ;
- Para isso, o escopo foi reduzido às notas que obedem a afinação igualmente temperada ($f_I = f_0 \cdot 2^{I/12}$), ou seja, notas com :
  - Note *On* ↔ Note *Of* ;
  - Note *On* ↔ Note *On* (*intensidade* = 0).
- As notas são armazenas numa lista encadeada, já com a frequência e amplitude determinadas, enquanto que a duração é determinada ao se encontrar o final ;

Aquisição Arquivo *Midi*

```java
    /**
     * Analisa a mensagem contida no arquivo midi
     * @param mensagem Mensagem atual do arquivo
     * @param instante Momento em que ocorre a mensagem
     */
    public static void tratarMensagem (MidiMessage mensagem, float instante){
        int IdOperacao = mensagem.getStatus();

        /*  Sendo "noteOn" ou "noteOff" */
        if ((128 <= IdOperacao) & (IdOperacao <= 159)){
            byte[] dados = mensagem.getMessage();
            NotasMidi notaAux = new NotasMidi();
            boolean noteOn;

            if (IdOperacao < 144)
                noteOn = false;
            else
                noteOn = true;

            if (noteOn & (dados[2] > 0)){
                notaAux.setFrequencia (dados[1]);
                notaAux.setAmplitude (dados[2]);
                notaAux.setInstanteInical (instante);

                notasLidas.add(notaAux);

            }else if (((noteOn & (dados[2] == 0))
                      || !noteOn) & !notasLidas.isEmpty()){
                notaAux.setFrequencia (dados[1]);

                int indice = localizarNota (notasLidas, notaAux);

                if (indice > -1){
                    notasLidas.get(indice).setInstanteFinal (instante);
                    notasLidas.get(indice).setDuracao();
                }
            }
        }
    }
```

Aquisição Arquivo *Midi*

- Para armazenar as notas registradas na melodia há a classe *NotasMidi*:

```java
/**
 * Responsavel por armazenar as notas lidas no arquivo midi,
 * antes de passa-las a uma classe Java
 * @author Helio Santana
 * @author Rodrigo Guimaraes
 * @version 1.0
 * @since   14/06/2016
 */
public class NotasMidi {
    /**
     * Frequencia base para as notas, obdecendo a afinacao igualmente temperada
     */
    public static final float f_0 = 8.175f;
    /**
     * Controle de amplitude da nota
     * Frequencia da nota, em Hz
     * Instantes limítrofes da nota
     * Duração da nota
     */
    int amplitude;
    float frequencia;
    float instanteInicial, instanteFinal;
    float duracao;
```

Conversão para *JavaClass*

- Com as notas determinadas, verifica-se a necessidade de preencher o começo da música com um vazio;
- A melodia formada é então escrita no final classe *BaseadoMidi.java*, criada de maneira *default* como uma classe sem melodias, como sendo um método próprio;
- Os métodos criados, as melodias, são adicionadas como sendo o penúltimo médodo da classe, pois o último é o método *gerenciador*.

Conversão para *JavaClass*

```
203     /**
204      * Escrita da melodia extraida do arquivo midi numa
205      * classe Java
206      * @param nomeClasse Nome da classe Java
207      */
208     public static void escreverArquivo (String nomeClasse){
209         if (!notasLidas.isEmpty()){
210             System.out.println("\tEscrevendo arquivo\n");
211 
212             try {
213                 File arq = new File (nomeClasse);
214 
215                 if (!arq.exists())
216                     arqNovo = true;
217 
218                 RandomAccessFile gravarArq = new RandomAccessFile(arq, "rw");
219 
220                 if (arqNovo){
221                     System.out.println("\t\tCriando arquivo das melodias\n");
222                     escreverCabecalho (gravarArq, nomeClasse);
223 
224                 }else{
225                     System.out.println("\t\tArquivo das melodias já existe\n");
226                     int desloca = posicionarSeek(arq, " public static Melodia gerenciador (String nome){");
227 
228                     if (desloca == -1)  desloca = 0;
229                     gravarArq.seek (desloca);
230                     gravarArq.writeBytes("\n");
231                 }
232 
233                 escreverMelodia (gravarArq, nomeArqMidiTratado);
234 
235                 arqNovo = false;
236                 gravarArq.close();
237             } catch (IOException e) {
238                 e.printStackTrace();
239             }
240 
241             System.out.println("\tAcabou de escrever\n");
242 
243         }else{
244             System.out.println("Não há notas a converter\n");
245         }
246     }
```

Conversão para *JavaClass*

- Obs: *amplitudeMaxNormalizada = 5.10f ;*

```
391     /**
392      * Escrita da melodia extraida do arquivo midi, em si
393      * @param arquivo Arquivo que contera as informacoes escritas
394      * @param nomeMetodo Nome do metodo a inserido no arquivo
395      */
396     public static void escreverMelodia (RandomAccessFile arquivo, String nomeMetodo){
397         try {
398             arquivo.writeBytes ("\n\tpublic static Melodia ");
399             arquivo.writeBytes (nomeMetodo);
400             arquivo.writeBytes (" (){\n");
401             arquivo.writeBytes ("\t\tnew BaseadoMidi();");
402             arquivo.writeBytes ("\n\n");
403 
404             for (int indNota = 0; indNota < notasLidas.size(); indNota++){
405                 NotasMidi notaAtual = notasLidas.get(indNota);
406                 float amplitude = ((amplitudeMaxNormalizada * notaAtual.getAmplitude()) / amplitudeMaxMidi);
407 
408                 arquivo.writeBytes("\t\tnota = new Nota (");
409                 arquivo.writeBytes(notaAtual.getDuracao() + ", ");
410                 arquivo.writeBytes(notaAtual.getFrequencia() + ", ");
411                 arquivo.writeBytes(amplitude + ");\n");
412                 arquivo.writeBytes("\t\tmelodiaPrincipal.addNota (nota);");
413                 arquivo.writeBytes("\n\n");
414             }
415 
416             arquivo.writeBytes("\n");
417             arquivo.writeBytes("\t\treturn melodiaPrincipal;\n");
418             arquivo.writeBytes("\t}\n");
419             arquivo.writeBytes("} ");
420         }catch (IOException e) {
421             e.printStackTrace();
422         }
423     }
```

Conversão para *JavaClass*

- Exemplo do método *gerenciador*:

```
5385    public static Melodia gerenciador (String nome){
5386        if (nome.equals("theRainsOfCastamere"))
5387            return theRainsOfCastamere();
5388        else if (nome.equals("bwv772"))
5389            return bwv772();
5390        else if (nome.equals("superMarioBros"))
5391            return superMarioBros();
5392        else return null;
5393    }
```

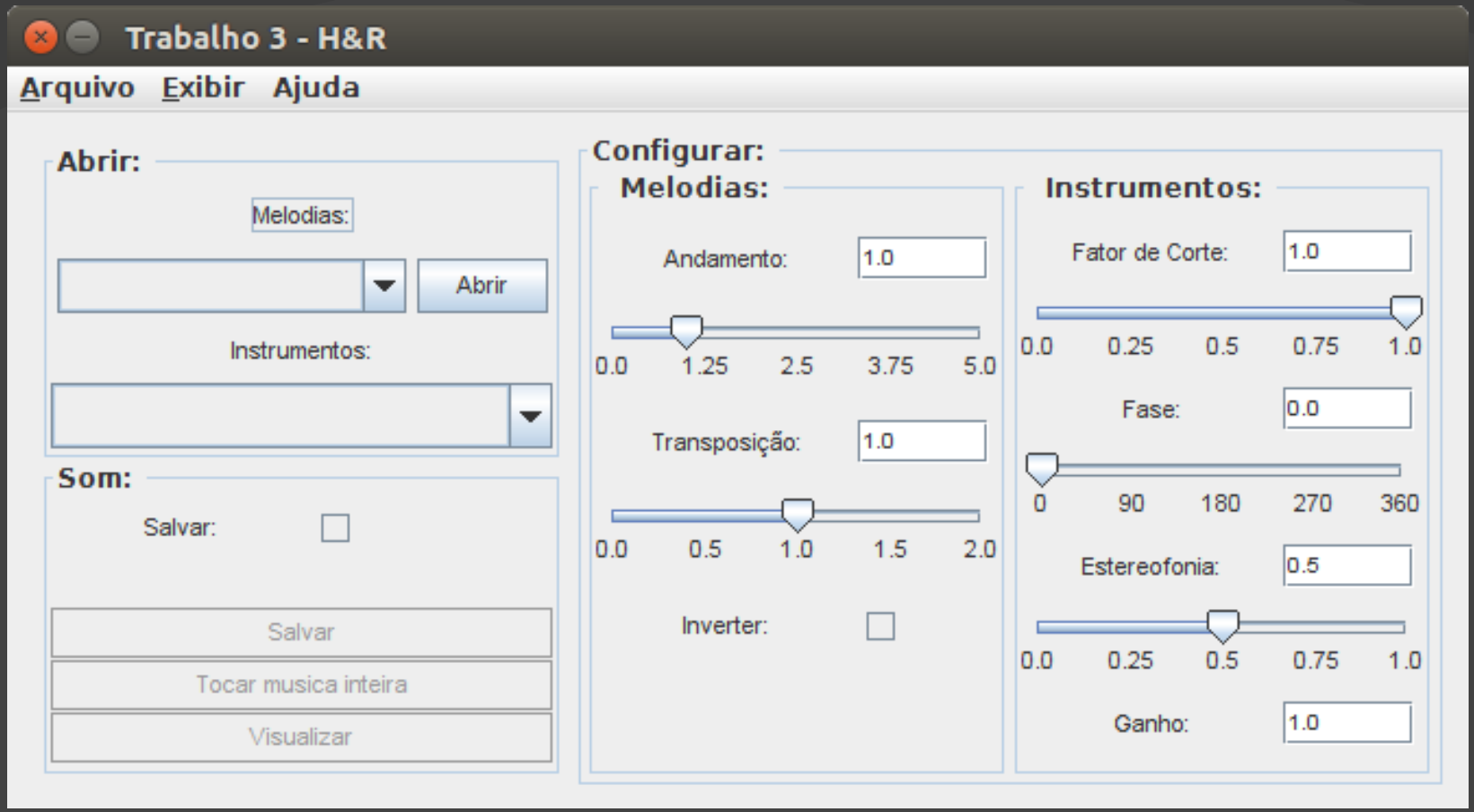
Conversão para *JavaClass*

```
24    public static Melodia theRainsOfCastamere (){
25        new BaseadoMidi();
26
27        nota = new Nota (2.0370934, 0.0, 0.0);
28        melodiaPrincipal.addNota (nota);
29
30        nota = new Nota (0.6772628, 219.9785, 3.2125983);
31        melodiaPrincipal.addNota (nota);
32
33        nota = new Nota (1.0167782, 349.19412, 3.2125983);
34        melodiaPrincipal.addNota (nota);
35
36        nota = new Nota (0.3377471, 219.9785, 3.2125983);
37        melodiaPrincipal.addNota (nota);
38
39        nota = new Nota (1.0167785, 329.59537, 3.2125983);
40        melodiaPrincipal.addNota (nota);
41
42        nota = new Nota (0.3377471, 219.9785, 3.2125983);
43        melodiaPrincipal.addNota (nota);
44
45        nota = new Nota (0.6772628, 349.19412, 3.2125983);
46        melodiaPrincipal.addNota (nota);
47
48        nota = new Nota (0.6772628, 391.95715, 3.2125983);
49        melodiaPrincipal.addNota (nota);
```

Interface Gráfica

- Funcionalidades :
  - Possibilidade de abrir um arquivo *Midi* para a conversão ;
  - Exibir informações sobre :
    - O conteúdo bruto do arquivo original, com algumas mensagens tratadas;
    - O conteúdo musical;
    - O conteúdo da *JavaClass* atual.
  - Controle sobre as melodias e os instrumento ;
  - Salvar, tocar ou visualizar o som formado.

Interface Gráfica



Repositório Online

- Todo o trabalho está disponível *online* no repositório gratuito *GitHub*, sob o link: https://github.com/rodrigofegui/ICS/tree/master/Trabalho3;
- Neste repositório estão contidos :
  - Arquivos fontes necessários para o trabalho;
  - Arquivos *Midi* básicos, para teste;
  - Arquivos de controle.

Referências

- **CIC – UnB**. *Package sintese*. Acessado em 2016. Disponível em: http://cic.unb.br/~lcmm/cic-icsonica-2016-1/ics/trabalhos/sintese/javadoc/index.html ;
- **Oracle – Docs**. *Java™ Platform, Standard Edition 7 API Specification*. Acessado em 2016. Disponível em: https://docs.oracle.com/javase/7/docs/api/;

Executando

- O programa construído terá suas funcionalidades demonstradas agora, comentários são bem-vindos.

FIM