# Licenciatura em Engenharia de Software

# Processamento de Linguagem

Daniel Amorim 11595

Tiago Gonçalves 13250

Ricardo Lopes 13677

# Conteúdo



# Introdução

O enunciado para a realização do trabalho prático que nós escolhemos foi o B. Linguagem Musical, isto por considerarmos o mais acessível, e que nos permitiria conjugar com os trabalhos das outras cadeira, apesar de ser acessível permite a compreensão dos recursos a serem utilizados jflex, byaccj e Java, que futuramente poderão representam uma ferramenta essencial para o nosso trabalho.

# Interpretação

## Inputs

O input é um arquivo .txt que apresenta uma notação caraterística da linguagem mid.

### Simples

.^, ._, .>, .<

Em que o símbolo precedente ao ponto representa um acréscimo ou decréscimo da variável em questão tempo ou velocidade;

### Complexos

.^{n}, ._{n}, .>{n}, .<{n}

Em que n representa a quantidade de vezes que o símbolo precedente ao ponto acresce ou decresce da variável em questão tempo ou velocidade;

## Identificação e Escrita

Recorremos á package kevinboone para escrita de cada nota.

## Output

O output do programa é um arquivo .mid que contem as notas do ficheiro input que foram convertidas pelo programa.

# Corpo do Programa

## JFLEX

Nesta parte do programa é feita a seleção do conteúdo legível para o programa, ou seja, os elementos que correspondam ás seguintes regras.

```
"."                         { return ParserTokens.DOT; }
"^"                         { return ParserTokens.UP; }
"^{"                        { return ParserTokens.UP_C; }
"_"                         { return ParserTokens.DOWN; }
"_{"                        { return ParserTokens.DOWN_C; }
">"                         { return ParserTokens.T_UP; }
">{"                        { return ParserTokens.T_UP_C; }
"<"                         { return ParserTokens.T_DOWN; }
"<{"                        { return ParserTokens.T_DOWN_C; }
"\MAJORSCALE"               { return ParserTokens.MAJOR;}

"\}"|"\{"   { return yytext().charAt(0); }

[0-9]+      {
                yyparser.yylval = new ParserVal(Integer.parseInt(yytext()));
                return ParserTokens.INT; }

[ \n\r]     { /* ignore */ }
.           { /* ignore */ }
```

## BYACC-J

Neste elemento é feito o redireccionamento da informação aceite para a parte de implementação estabelece uma comunicação entre o Jflex e o JAVA.

```
%token INT DOT UP UP_C DOWN DOWN_C T_UP T_UP_C T_DOWN T_DOWN_C MAJOR
%%
        axiom: program    { commands = (ArrayList<Command>) $1.obj ; }
;

program: /* epsilon */   {
$$ = new ParserVal(new ArrayList<Command>());
}
|  program instruction { ((ArrayList<Command>) $1.obj ).add( (Command)$2.obj );  $$ = $1; }
;

instruction: DOT { $$ = new ParserVal(new Dot()); }
            | UP_C value '}' { $$ = new ParserVal(new Up((Value)$3.obj)); }
            | UP { $$ = new ParserVal(new Up()); }
            | DOWN_C value '}' { $$ = new ParserVal(new Down((Value)$3.obj)); }
            | DOWN { $$ = new ParserVal(new Down()); }
            | T_UP { $$ = new ParserVal(new Tempo_Increase()); }
            | T_UP_C value '}' { $$ = new ParserVal(new Tempo_Increase_C($3.ival)); }
            | T_DOWN { $$ = new ParserVal(new Tempo_Decrease()); }
            | T_DOWN_C value '}' { $$ = new ParserVal(new Tempo_Decrease_C($3.ival)); }
            ;

value: INT  {  $$ = new ParserVal(new Value($1.ival)); }
;
```

## JAVA

### Main

O programa é executado.

```
public static void main(String args[])
{
    Parser parser = null;
    try
    {
        parser = new Parser(
                new InputStreamReader(
                        new FileInputStream("src/main/resources/input.txt")
                ));
    }
    catch(FileNotFoundException e){
        System.err.println("File input not found!");
        System.exit(1);
    }

    parser.yyparse();

    System.out.println();

    Note note = new Note();
    MidiFile mf = new MidiFile();

    PrintStream file = null;
    try {
        file = new PrintStream("target/out.txt");
    } catch (FileNotFoundException e) {
        System.err.println("Erro de criação");
        System.exit(1);
    }

    for (Command c: parser.commands) {
        note = c.execute(note, mf, file);
    }
    try {
        mf.writeToFile("MUSIC.mid");
    }
    catch (Exception e)
    {
        System.err.println("O ficheiro não foi criado.");
    }
}
```

## Class Note

Estrutura para receção da informação da cada nota.

```
public class Note {
    int nota, velocidade;
    VarTab vars;

    public Note() {
        nota = 60; // dó
        velocidade = 32;
        vars = new VarTab();
```

```
    }

    public void AumentaVelocidade()
    {
        if(velocidade / 2 >= 4)
            velocidade = velocidade / 2;
        else
            velocidade = 4;
    }

    public void DiminuiVelocidade()
    {

        if(velocidade * 2 <= 64)
            velocidade = velocidade *2;
        else
            velocidade = 64;
    }

    public void DiminuiVelocidadeC(int add)
    {
        int aux;
        for(int i= 0;i<add;i++)
        {
            if (velocidade==64)
            {
                i = add;
                break;
            }
            aux = velocidade;
            aux *= 2;
            velocidade = aux;
        }
    }

    public void AumentaVelocidadeC(int add)
    {
        int aux;
        for(int i= 0;i<add;i++)
        {
            if (velocidade==4)
            {
                i = add;
                break;
            }
            aux = velocidade;
            aux /= 2;
            velocidade = aux;
        }
    }
}
```

## MidiFile

Estrutura definida pelo autor da ideia original de conversão digital das notas.

http://www.kevinboone.net/javamidi.html

## Comandos

Classe abstrata para receber os comandos específicos.

```
import java.io.PrintStream;
public abstract class Command {
    public abstract Note execute(Note n, MidiFile mf, PrintStream s);
}
```

## Up

- Aumenta meio tom.
- Aumento meio tom as vezes determinadas pelo utilizador.

```
public class Up extends Command {
    private Value repeatTimes;

    public Up() {}
    public Up(Value u) {repeatTimes = u;}

    public Note execute(Note n, MidiFile mf, PrintStream s) {
        if (repeatTimes == null) {
            n.nota++;
            return n;
        }

        for (int i=0; i < repeatTimes.Eval(n); i++)
            n.nota++;

        return n;
    }
}
```

## Down

- Desce meio tom.
- Desce meio tom as vezes determinadas pelo utilizador

```
public class Down extends Command {
    private Value repeatTimes;

    public Down() {}
    public Down(Value u) {repeatTimes = u;}

    public Note execute(Note n, MidiFile mf, PrintStream s) {
        if (repeatTimes == null) {
            n.nota--;
            return n;
        }

        for (int i=0; i < repeatTimes.Eval(n); i++)
            n.nota--;

        return n;
    }
}
```

## VelocityUp

- Aumenta em um tempo a velocidade da nota.

```
public class Tempo_Increase extends Command {

    public Note execute(Note n, MidiFile mf, PrintStream s) {
        n.AumentaVelocidade();
        return n;
```

```
    }
}
```

- Aumenta mais que uma vez.

```
public class Tempo_Increase_C extends Command {

    private int add;

    public Tempo_Increase_C(int u) {add = u;}

    public Note execute(Note n, MidiFile mf, PrintStream s) {
        n.AumentaVelocidadeC(add);
        return n;
    }
}
```

## VelocityDown

- Diminui em um tempo a velocidade da nota.

```
public class Tempo_Decrease extends Command {

    public Note execute(Note n, MidiFile mf, PrintStream s) {
        n.DiminuiVelocidade();
        return n;
    }
}
```

- Diminui mais que uma vez.

```
public class Tempo_Decrease_C extends Command {

    private int add;

    public Tempo_Decrease_C(int u) {add = u;}

    public Note execute(Note n, MidiFile mf, PrintStream s) {
        n.DiminuiVelocidadeC(add);
        return n;
    }
}
```

## VarTab

Mantém o registo em memoria das notas.

```
public class VarTab {

    private Hashtable<String, Integer> dic;

    public  VarTab() {
        dic = new Hashtable<String, Integer>();
    }
```

```
    public void setVariable(String var, int val) {
        dic.put(var, val);
    }

    public int getVariable(String var) throws Exception {
        if (dic.containsKey(var)) {
            return dic.get(var);
        }
        throw new Exception("Variable "+ var +" not defined");
    }
}
```

### Value

Extrai a informação da Hastable.

```
public class Value {
    private int intVal;
    private String varName;

    public Value(String n) { varName = n; }
    public Value(int v) { intVal = v; }

    public int Eval(Note n) {
        if (varName == null) {
            return intVal;
        }

        try {
            return n.vars.getVariable(varName);
        } catch (Exception e) {
            System.err.println(e);
            System.exit(1);
        }

        return -1;
    }
}
```

## Conclusão

A realização deste trabalho permitiu-nos adquirir conhecimento sobre as ferramentas utilizadas que para certos problemas correspondem ao método mais eficaz de desenvolvimento

## Bibliografia


http://www.kevinboone.net/javamidi.html

https://elearning.ipca.pt/1819/course/view.php?id=8360