

Marmalade

Raphael Norwitz

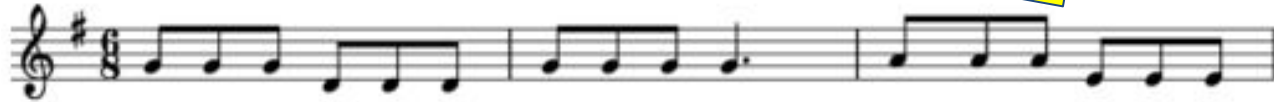
Savvas Petridis

Cathy Jin

Uzo Amuzie

Motivation

```
import jm.JMC;  
import jm.music.data.*;  
import jm.util.*;  
import jm.audio.*;  
  
public final class SonOfBing implements JMC{  
  
    public static void main(String[] args){  
  
        Score score = new Score(new Part(new Phrase(new Note(C4, MINIM))));  
        Write.midi(score);  
        Instrument inst = new SawtoothInst(44100);  
        Write.au(score, inst);  
  
    }  
}
```





phrase p = \$(PIANO) [

\$(6:8) [67.e, 67.e, 67.e, 62.e, 62.e, 62.e],

\$(6:8) [67.e, 67.e, 67.e, 67.h],

\$(6:8) [69.e, 69.e, 69.e, 64.e, 64.e, 64.e],

];

Result: Song Creation

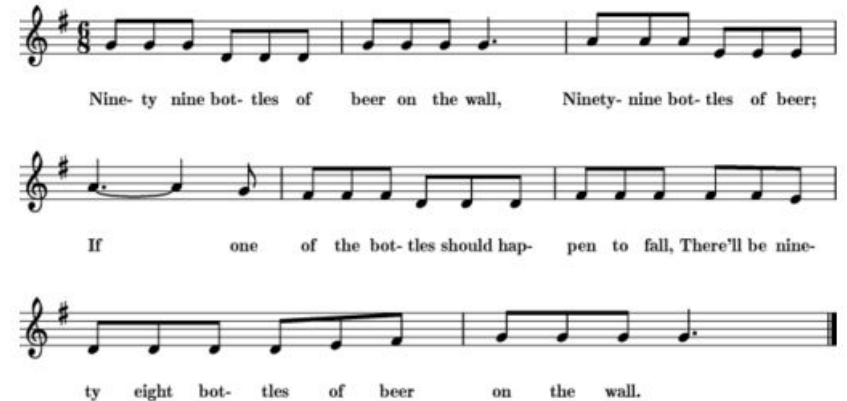
```
/* 99 bottles of beer */
```

```
measure t_1 = $(6:8) [67.e, 67.e, 67.e, 62.e, 62.e, 62.e];  
measure t_2 = $(6:8) [67.e, 67.e, 67.e, 67.h];  
measure t_3 = $(6:8) [69.e, 69.e, 69.e, 64.e, 64.e, 64.e];  
measure t_4 = $(6:8) [69.h, 0.e, 0.e, 67.e];  
measure t_5 = $(6:8) [65.e, 65.e, 65.e, 62.e, 62.e, 62.e];  
measure t_6 = $(6:8) [65.e, 65.e, 65.e, 65.e, 65.e, 64.e];  
measure t_7 = $(6:8) [62.e, 62.e, 62.e, 62.e, 64.e, 65.e];  
measure t_8 = $(6:8) [67.e, 67.e, 67.e, 67.h];
```

```
/* make phrase associated with 'instrument' */  
phrase ph1 = $(HARP) [t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8];  
phrase ph2 = $(HARP) [ t_2, t_3, t_4, t_5, t_6, t_7, t_8 ];
```

```
/* put phrases into song, assign bpm */  
song s1 = $(60) [ph1];
```

```
(play(), write()) [s1, s1];
```



The image displays three staves of musical notation in treble clef with a key signature of one sharp (F#) and a 6/8 time signature. The first staff contains the melody for the first line of the song: "Nine- ty nine bot- tles of beer on the wall, Ninety- nine bot- tles of beer;". The second staff contains the melody for the second line: "If one of the bot- tles should hap- pen to fall, There'll be nine-". The third staff contains the melody for the third line: "ty eight bot- tles of beer on the wall." The lyrics are written below the notes, with hyphens indicating syllables that span across multiple notes.

Result: Imperative Programs

```
/* 99 bottles of beer in marmalade */
```

```
int offset = 0;
```

```
int current_bottle = 99;
```

```
int next_bottle = 98;
```

```
while(offset < 98)
```

```
{
```

```
    current_bottle = current_bottle - 1;
```

```
    next_bottle = next_bottle - 1;
```

```
    (print(), print()) [ current_bottle, " bottles of beer on the wall " ];
```

```
    (print(), print()) [ current_bottle, " bottles of beer. Take one down, pass it around, "];
```

```
    (print(), print()) [ next_bottle, " bottles of beer on the wall."]
```

```
    offset = offset + 1;
```

```
}
```

Result: Algorithms

```
/* calculate nth fibonacci number */
```

```
funk int int fib(int n, int val_1, int val_2)
{
    if(n <= 2){
        return 1;
    }
    else{
        val_1 = $fib(n-1, 0, 0);
        val_2 = $fib(n-2, 0, 0);
        n = val_1 + val_2;
        return n;
    }
}
```

```
/* gcd algorithm */
```

```
funk int int gcd(int a, int b)
{
    while(a != b){
        if (a > b)
            { a = a - b; }
        else
            { b = b - a; }
    }
    return a;
}
```

```
(print(), print()) [$gcd(30, 90), $fib(10, 0, 0)];
```

```
>> ./test_gcd_fib
```

```
30
```

```
55
```

Result: Function Application

```
/* different instruments */
```

```
( print(), play(), print(), play() ) [ "Piano:", $(PIANO) [t_1, t_2], "Honkytonk:", $(HONKYTONK) [t_1, t_2] ];
```

```
/* order */
```

```
( play(), play(), play() ) [ $(PIANO) [t_1, t_2, t_3], $(PIANO) [t_3, t_1, 2], $(PIANO) [t_3, t_8, t_4] ];
```

```
/* tempo */
```

```
(play(), play(), play() ) [$ (60) [ph1, ph2], $(30) [ ph1, ph2 ], $(120) [ph1, ph2] ];
```

```
/* function application */
```

```
( play(), play(), play() ) [ph1, $transpose_phrase(ph1, 5), $transpose_measure(t_1, 7) ];
```

Result: Algorithmic Modification

```
funk measure measure transpose_measure_w(measure m, int n, int counter, int  
j, note k, measure l)
```

```
{
```

```
    j = $length_measure(m);
```

```
    counter = 0;
```

```
    l = $evaluate_measure(m);
```

```
    while(counter < j)
```

```
    {
```

```
        k = l&counter;
```

```
        l&counter = k + n;
```

```
        counter = counter + 1;
```

```
    }
```

```
    return l;
```

```
}
```

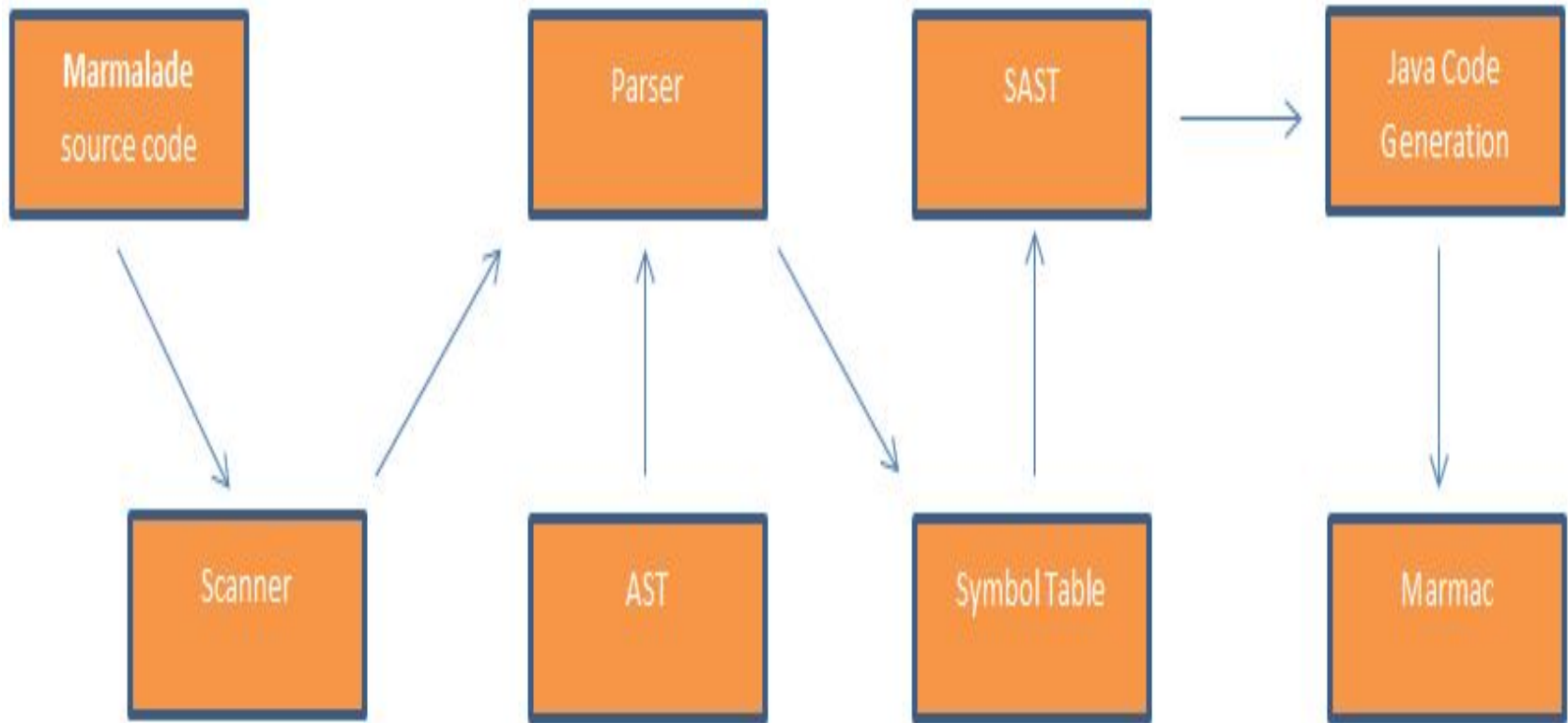
```
funk measure measure transpose_measure(measure m, int n)
```

```
{
```

```
    return $transpose_measure_w(m, n, 0, 0, 44.q, $() [55.h]);
```

```
}
```


Overall Structure



Java Implementation

- jMusic library to implement in Java
- marmalade.jar package
 - Custom library
 - Streamline and unify function calls

Lessons Learned

- Develop core features of the language first!
 - Append/Regex
- Limited by jMusic
- Importance of team communication & starting project early

Demo Time!!



YUM!!!