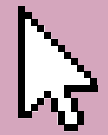
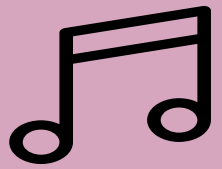
Music Builder:

A Computer Science Project



By Rani Patel

The JFugue Library



```
1 import org.jfugue.pattern.Pattern;
2 import org.jfugue.player.Player;
3 import org.jfugue.theory.Note;
4 import org.jfugue.theory.ChordProgression;
5 import java.lang.Math;
6 import org.jfugue.theory.Chord;
```

Important classes:

- Player class
- Pattern class
- ChordProgression class
- Chord class
- Note class

ChordProgression

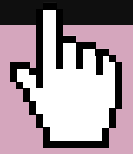
```
public static ChordProgression randMajTonic(){
   int num = (int)(Math.random() * 9);
   ChordProgression cp = new ChordProgression("");
   if(num <= 3){
      cp = new ChordProgression("I");
   }
   if(num == 4 || num == 5){
      cp = new ChordProgression("iii");
   }
   if(num == 6 || num == 7){
      cp = new ChordProgression("vi");
   }
   if(num == 8){
      cp = randMaj();
   }
   return cp;
}
```

RandomChords Class:

- static methods
- different methods for tonic, predominant, and dominant
- uses math.random to control probability

Patterns (Part 1)

```
31    public static String randEighthNote(){
32       int num = (int)(Math.random() * 4);
33       if(num == 0){
34          return "$0i ";
35       }
36       if(num == 1){
37          return "$1i ";
38       }
39       if(num == 2){
40          return "$2i ";
41       }
42       if(num == 3){
43          return "Ri ";
44       }
45       return "";
46    }
```



Sequence Class:

- creates the basic string needed to build patterns
- creates a root, third, fifth, and rest note

Patterns (Part 2)

```
public static String randHalfBeat(int i){
   int num = (int)(Math.random() * 8);
   if(i == 0){
      if(num == 0){
         return randHalfNote();
      }
      if(num == 1 || num == 2 || num == 3){
         return randQuaterBeat(i) + randQuaterBeat(i);
      }
      if(num == 4 || num == 5){
         return randDotQuaterBeat(i) + randEighthBeat(i);
      }
      if(num == 6 || num == 7){
         return randEighthBeat(i) + randDotQuaterBeat(i);
      }
   }
   else{
      if(num == 0 || num == 1 || num == 2){
         return randHalfNote();
      }
      if(num == 3 || num == 4){
         return randQuaterBeat(i) + randQuaterBeat(i);
      }
      if(num == 5){
         return randDotQuaterBeat(i) + randEighthBeat(i);
      }
      if(num == 6 || num == 7){
         return randEighthBeat(i) + randDotQuaterBeat(i);
      }
   }
```

Sequence Class:

- makes the beats of the music using the previous note-builder methods
- the input decides whether the ending notes should be longer or not

Phrases

```
public static Pattern[] smallMajFourFour(boolean e, String k){
   //for chords
   Pattern p1 = new Pattern("V0");
   //for sequences
   Pattern p2 = new Pattern("V1");
   ChordProgression cp;
   Pattern[] p4 = {p1, p2};
   int num = (int)(Math.random() * 2);
   //one chord in tonic
   if(num == 0){
      cp = new ChordProgression("I").setKey(k);
      p1.add(cp.eachChordAs(Sequences.getChord(cp, "w")));
      p2.add(cp.eachChordAs(Sequences.randWholeBeat(0)));
   }
   //two chords in tonic
   if(num == 1){
      cp = new ChordProgression("I").setKey(k);
      p1.add(cp.eachChordAs(Sequences.getChord(cp, "h")));
      p2.add(cp.eachChordAs(Sequences.randHalfBeat(0)));
      cp = RandomChord.randMajTonic().setKey(k);
      p1.add(cp.eachChordAs(Sequences.getChord(cp, "h")));
      p2.add(cp.eachChordAs(Sequences.randHalfBeat(0)));
   }
   num = (int)(Math.random() * 2);
   //two chords in predom
   if(num == 0){
      for(int i = 0; i < 2; i++){
         cp = RandomChord.randMajPreDom().setKey(k);
         p1.add(cp.eachChordAs(Sequences.getChord(cp, "w")));
         p2.add(cp.eachChordAs(Sequences.randWholeBeat(0)));
```

Phrase Class:

- combines the Sequence class and RandomChord class
- creates a Pattern array that can be played with one part being the chords and the other the melody

Music Builder (Part 1)

```
11 public class RandomMusic{
12    public static void main(String[] args) throws IOException{
13 
14          Scanner scanner = new Scanner(System.in);
15 
16          System.out.println("Enter a key (major or minor): ");
17          //true is major, false is minor
18          Boolean mode = true;
19          String key = scanner.next() + scanner.next();
20          if(key.indexOf("minor") != -1 ){
21             mode = false;
22          }
23          int space = key.indexOf("m");
24          key = key.substring(0, space);
25 
26          System.out.println("Which time signature: 4/4, 3/4, or 6/8: ");
27          String sign = scanner.next();
28 
29          System.out.println("What length: short, medium, long: ");
30          String length = scanner.next();
31 
32          System.out.println("Enter a tempo: ");
33          int tempo = scanner.nextInt();
34 
35          scanner.close();
36 
37          Pattern p1 = new Pattern();
38          Pattern p2 = new Pattern();
39          Pattern[] p3 = {p1, p2};
40          Player player = new Player();
41 
```

RandomMusic Class:

- uses the scanner class to receive input
- user can decide between the meter, key, tempo, and length

Music Builder (Part 2)

```
if(length.equals("short")){
   if(mode){
      if(sign.equals("4/4")){
         p3 = Phrase.smallMajFourFour(true, key);
      }
      if(sign.equals("3/4")){
         p3 = Phrase.smallMajThreeFour(true, key);
      }
      if(sign.equals("6/8")){
         p3 = Phrase.smallMajSixEight(true, key);
      }
   }
   else{
      if(sign.equals("4/4")){
         p3 = Phrase.smallMinFourFour(true, key);
      }
      if(sign.equals("3/4")){
         p3 = Phrase.smallMinThreeFour(true, key);
      }
      if(sign.equals("6/8")){
         p3 = Phrase.smallMinSixEight(true, key);
      }
   }
}
```

RandomMusic Class:

- calls the method necessary according to the input provided

Music Builder (Part 3)

```
97          }
98          p1 = p3[0];
99          p2 = p3[1];
100
101         p1.setTempo(tempo);
102         p2.setTempo(tempo);
103
104         System.out.println(p1);
105         System.out.println(p2);
106
107         player.play(p1, p2);
108     }
109 }
```

RandomMusic Class:

- sets the tempo
- prints out the chords + melody pattern
- plays the music