



MIDI: programmare e comporre



Blanco Francesco Giulio



Indice breve

1. Teoria Musicale
2. Brano in Java
3. Brano in Finale
4. Java → Finale





Indice completo



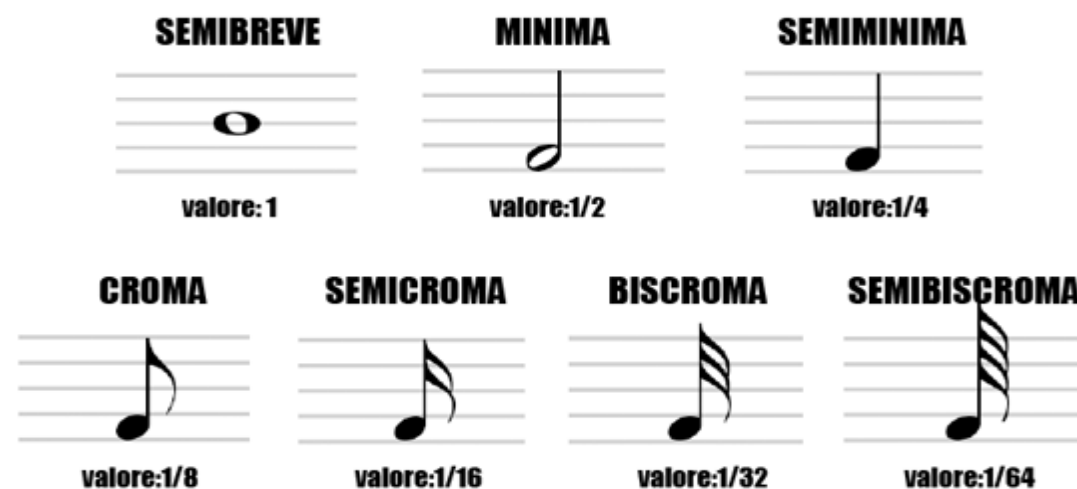
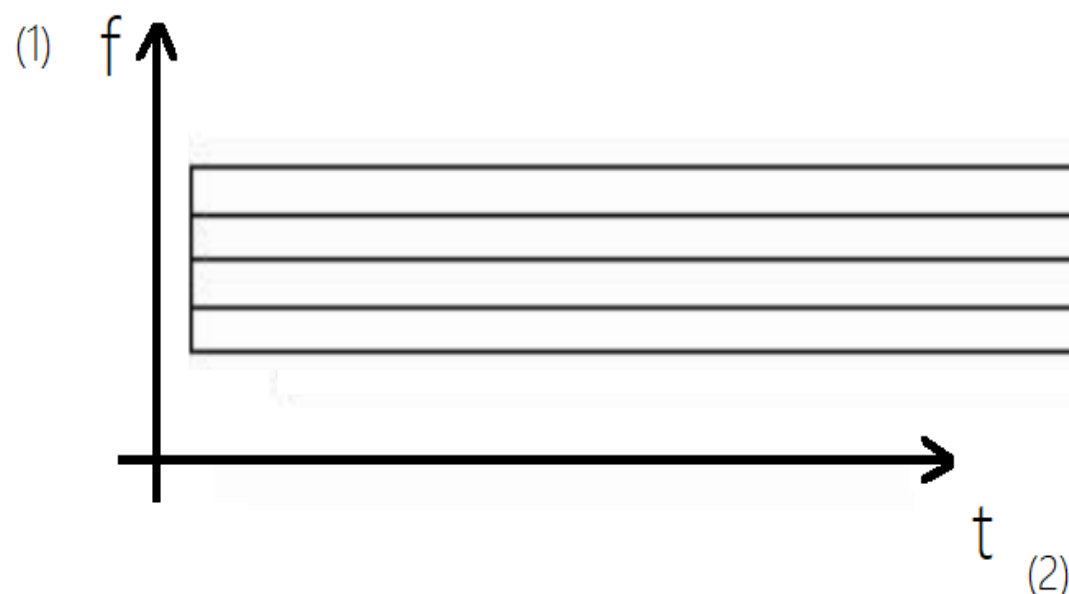
1. Introduzione
 - ❑ Elementi di Teoria Musicale
2. Imparare a scrivere brani musicali in Java
 - ❑ Il protocollo MIDI con approccio top-down
 - **Scriviamo un brano musicale in Java**
3. Un nuovo strumento compositivo: Finale
 - ❑ **Scriviamo un brano musicale in Finale**
4. I risultati del progetto
 1. La scarsa scalabilità del codice Java
 2. Il crollo della complessità su Finale
 3. **Java → Finale** [*pipeline concettuale*]





1. Teoria Musicale

- Le note musicali sono segni grafici mirati a rappresentare concettualmente e graficamente dei suoni (toni puri)
- Posso rappresentare le note in un “grafico” detto rigo musicale
- Il simbolo di una nota denota la sua durata





2. Brano in Java (1)

- Sequencer: decide la sequenza di eventi che devono essere eseguiti e i rispettivi parametri
 1. Sequencer sequencer = getSequencer();
 2. sequencer.setTempoInBPM(164.0f);
 3. Sequence seq = **getMidiInputData()**;
 - **setMidiEvents()**;
 - **addMidiEvent()**;
 - setMessage(); **[DOMANDA]**
 4. sequencer.setSequence(seq);
 5. sequencer.start();
 6. **salvaSuFile(seq)**;
 7. sequencer.close();

Nota: analizzeremo i metodi in grassetto
(non forniti dalla libreria)



2. Brano in Java (2)

```
private Sequence getMidiInputData() {  
    int ticksPerQuarterNote = 4;  
    Sequence seq;  
    try {  
        seq = new Sequence(Sequence.PPQ, ticksPerQuarterNote);  
        setMidiEvents(seq.createTrack());  
    } catch (InvalidMidiDataException e) {  
        e.printStackTrace();  
        return null;  
    }  
    return seq;  
}
```




2. Brano in Java (3)

```
private void setMidiEvents(Track track) {
    ShortMessage sm = new ShortMessage( );
    int instrument = 26;
    sm.setMessage(ShortMessage.PROGRAM_CHANGE,
2, instrument, 0);
    track.add(new MidiEvent(sm, 0));

    int channel = 1;
    int velocity = 64;
    int note = 60;
    int tick = 0;

addMidiEvent(track, ShortMessage.NOTE_ON, channel, note, velocity, tick);
    tick+=3;
addMidiEvent(track, ShortMessage.NOTE_OFF, channel, note, 0, tick);
    tick+=1;
    [addMidiEvent...]
}
```



Frequency	Keyboard	Note name	MIDI number
4186.0		C8	108
3951.1		B7	107
3729.3		A7	106
3520.0		G7	105
3322.4		F7	104
3136.0		E7	103
2960.0		D7	102
2793.8		C7	101
2637.0		B6	100
2489.0		A6	99
2349.3		G6	98
2217.5		F6	97
2093.0		E6	96
1975.5		D6	95
1864.7		C6	94
1760.0		B5	93
1661.2		A5	92
1568.0		G5	91
1480.0		F5	90
1396.9		E5	89
1318.5		D5	88
1244.5		C5	87
1174.7		B4	86
1108.7		A4	85
1046.5		G4	84
987.77		F4	83
932.33		E4	82
880.00		D4	81
830.61		C4	80
783.99		B3	79
739.99		A3	78
698.46		G3	77
659.26		F3	76
622.25		E3	75
587.33		D3	74
554.37		C3	73
523.25		B2	72
493.88		A2	71
466.16		G2	70
440.0		F2	69
415.30		E2	68
392.00		D2	67
369.99		C2	66
349.23		B1	65
329.63		A1	64
311.13		G1	63
293.67		F1	62
277.18		E1	61
261.6		D1	60
246.94		C1	59
233.08		B0	58
220.00		A0	57
207.65			56
196.00			55
185.00			54
174.61			53
164.81			52
155.56			51
146.83			50
138.59			49
130.81			48
123.47			47
116.54			46
110.00			45
103.83			44
97.999			43
92.499			42
87.307			41
82.407			40
77.782			39
73.416			38
69.296			37
65.406			36
61.735			35
58.270			34
55.000			33
51.913			32
48.999			31
46.249			30
43.654			29
41.203			28
38.891			27
36.708			26
34.648			25
32.703			24
30.868			23
29.135			22
27.500			21



2. Brano in Java (4)

```
private void addMidiEvent(Track track, int command, int channel, int
data1, int data2, int tick) {
    ShortMessage message = new ShortMessage();
    try {
        message.setMessage(command, channel, data1, data2);
    } catch (InvalidMidiDataException e) {
        e.printStackTrace();
    }
    track.add(new MidiEvent(message, tick));
}
```

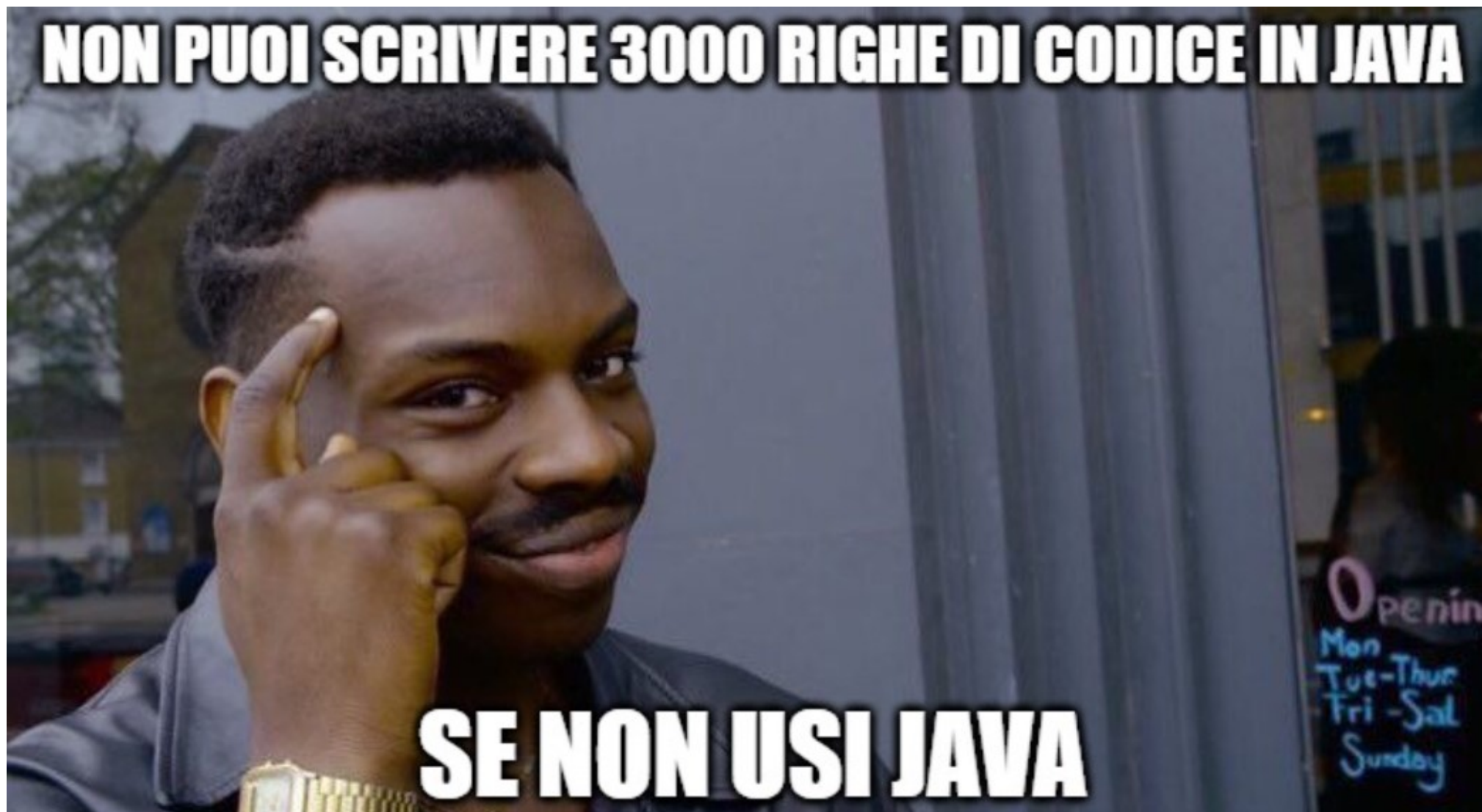



2. Brano in Java (5)

```
private void salvaSuFile(Sequence seq) {  
    File f = new File("midifile.mid");  
    try {  
        MidiSystem.write(seq,1,f);  
    } catch (IOException ex) {  
        ex.printStackTrace();  
    }  
    System.out.println("midifile end ");  
}
```



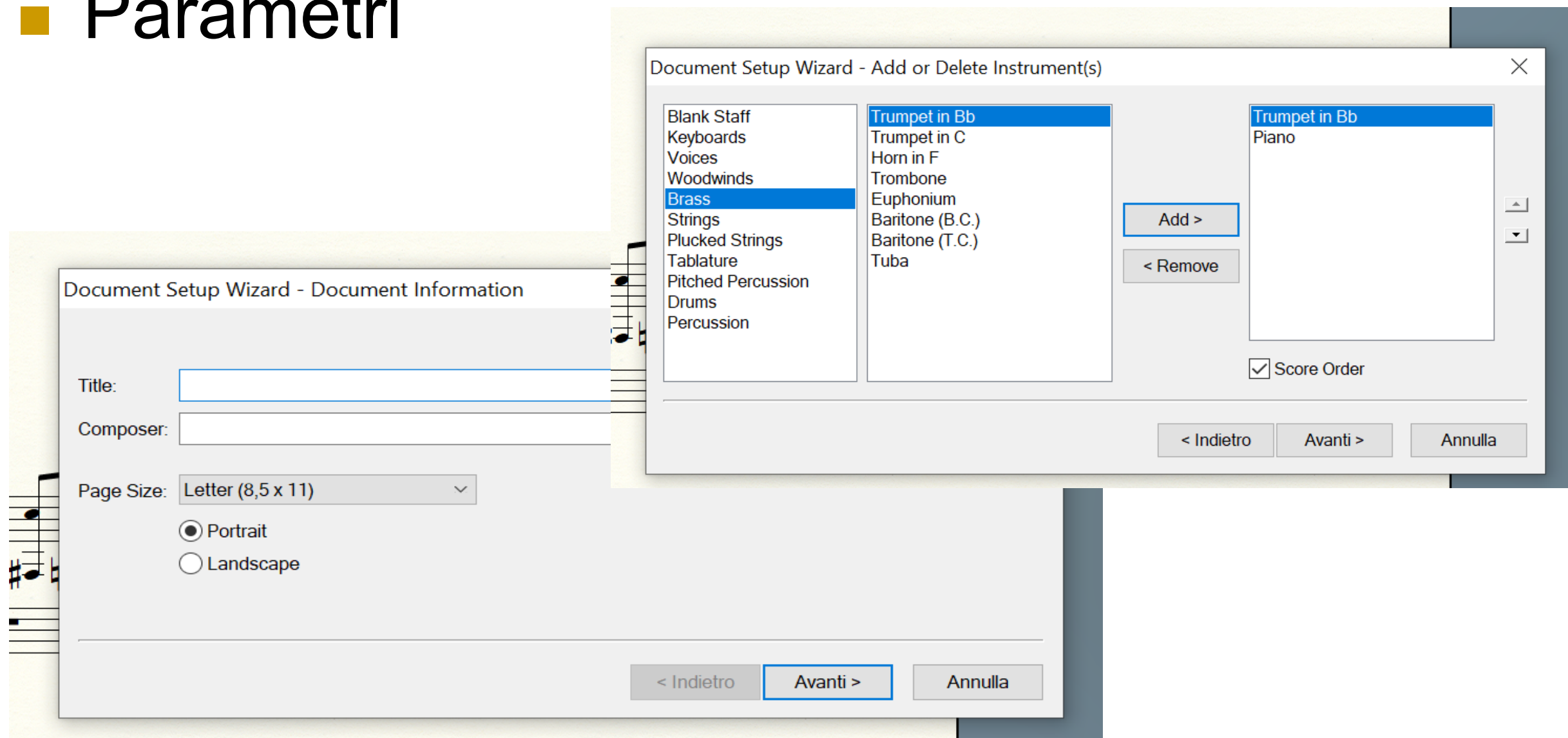
2.1 Complessità crescente





3. Brano in Finale (1)

■ Parametri





3. Brano in Finale (2)

■ Partitura

Tico Tico

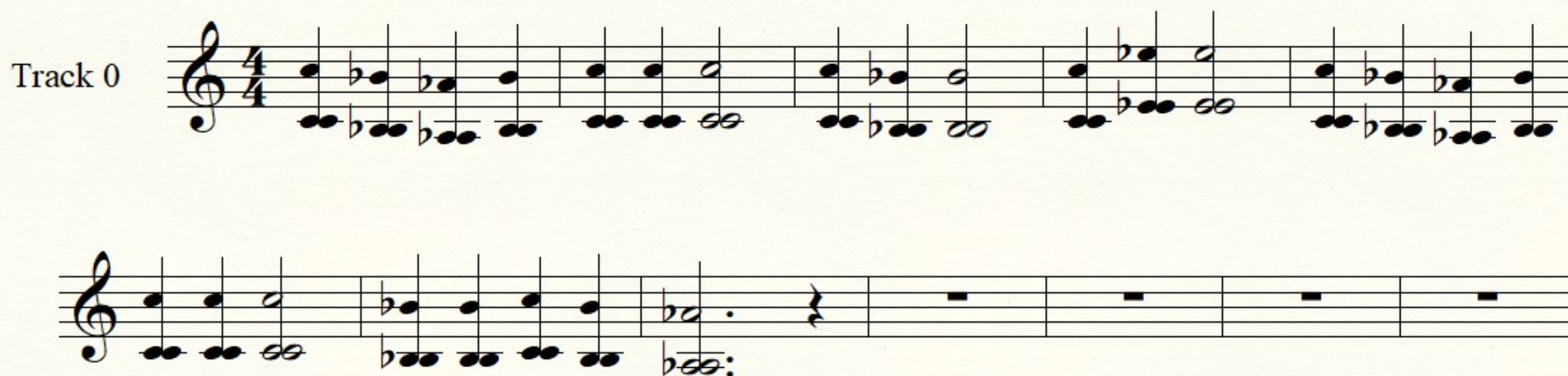
Duo chitarre



4. Java → Finale

- Differenze Java-Finale
- L'output dei due programmi è identico
- *Pipeline concettuale*: posso aprire il file output di Java con Finale

Java MIDIfile, Francesco Giulio Blanco





Domanda 1

- Se volessi scrivere in Java un programma che suoni N note, quante volte (almeno) dovrei chiamare il metodo `setMessage()` (libreria `javax.sound.midi`)?
 - A. N
 - B. $N+2$
 - C. $2N+2$
 - D. $2N$
 - E. $N-2$



Domanda 1

- Se volessi scrivere in Java un programma che suoni N note, quante volte (almeno) dovrei chiamare il metodo `setMessage()` (libreria `javax.sound.midi`)?
 - A. N
 - B. $N+2$
 - C. $2N+2$
 - D. **$2N$**
 - E. $N-2$



Domanda 2

- Sia dato il seguente codice Java (javax.sound.midi):

```
ShortMessage sm = new ShortMessage();
```

```
int instrument = 26;
```

```
[COMPLETE]
```

```
track.add(new MidiEvent(sm, 0));
```

- Completare la riga [COMPLETE]:

- A. `sm.setMessage(ShortMessage.PROGRAM_CHANGE, 0);`
- B. `sm.setMessage(ShortMessage.PROGRAM_CHANGE, 2, instrument, 0);`
- C. `sm.setMessage(LongMessage.PROGRAM_CHANGE, 2, instrument, 0);`
- D. `sm.setMessage(LongMessage.CHANGE_INSTR, 0);`
- E. `sm.setMessage(ShortMessage.CHANGE_INSTR, 2, instrument, 0);`
- F. `sm.setMessage(ShortMessage.CHANGE_INSTR, 0);`



Domanda 2

- Sia dato il seguente codice Java (javax.sound.midi):

```
ShortMessage sm = new ShortMessage();
```

```
int instrument = 26;
```

```
[COMPLETE]
```

```
track.add(new MidiEvent(sm, 0));
```

- Completare la riga [COMPLETE]:

- A. `sm.setMessage(ShortMessage.PROGRAM_CHANGE, 0);`
- B. `sm.setMessage(ShortMessage.PROGRAM_CHANGE, 2, instrument, 0);`
- C. `sm.setMessage(LongMessage.PROGRAM_CHANGE, 2, instrument, 0);`
- D. `sm.setMessage(LongMessage.CHANGE_INSTR, 0);`
- E. `sm.setMessage(ShortMessage.CHANGE_INSTR, 2, instrument, 0);`
- F. `sm.setMessage(ShortMessage.CHANGE_INSTR, 0);`



Conclusioni

- Spesso la strada più difficile è quella più giusta. Questo principio non vale nell'informatica ;)





Contatti

Francesco Giulio Blanco

Dip. di Ing. Inf., III anno

Email Istituzionale:

blanco.francesco@studium.unict.it



UNIVERSITÀ
degli STUDI
di CATANIA

GRAZIE PER L'ATTENZIONE