MIDI: programmare e comporre

Blanco Francesco Giulio

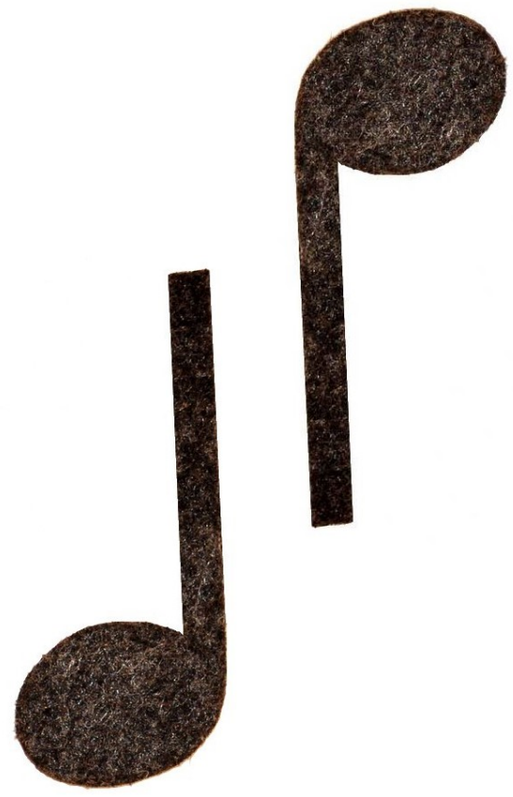
Progetti
Informatica Musicale 2020/21

Indice breve

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

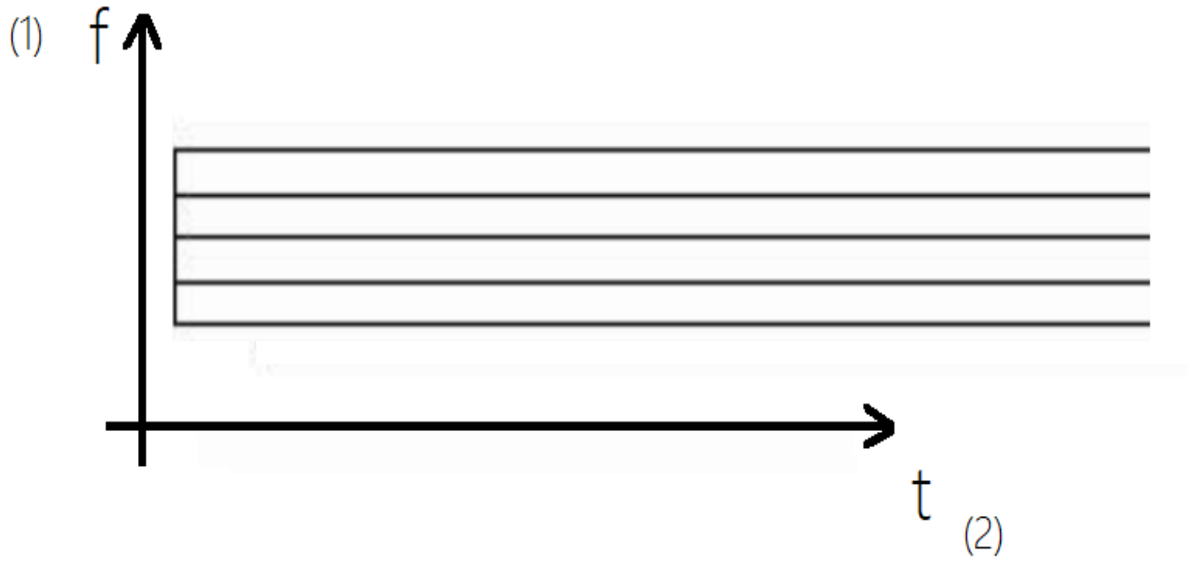
Indice completo

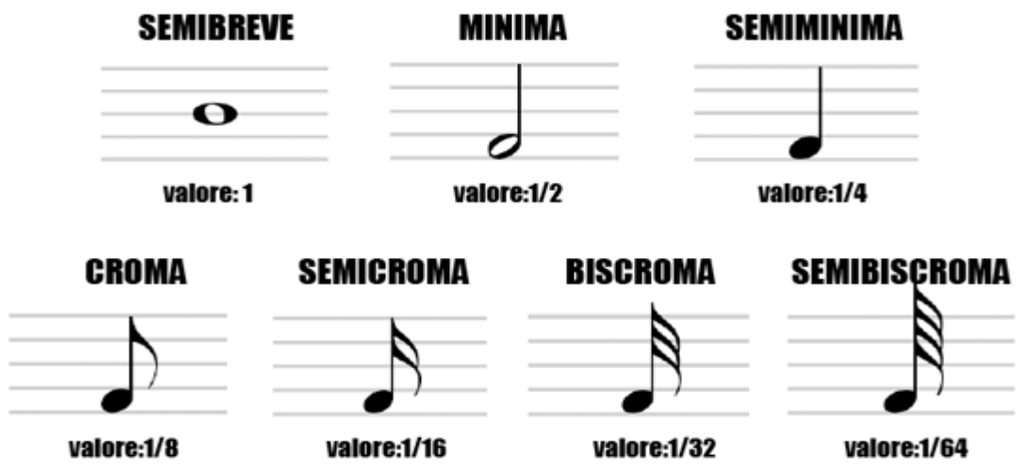
1. Introduzione
   - Elementi di Teoria Musicale
2. Imparare a scrivere brani musicali in Java
   - Il protocollo MIDI con approccio top-down
     - **Scriviamo un brano musicale in Java**
3. Un nuovo strumento compositivo: Finale
   - **Scriviamo un brano musicale in Finale**
4. I risultati del progetto
   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)

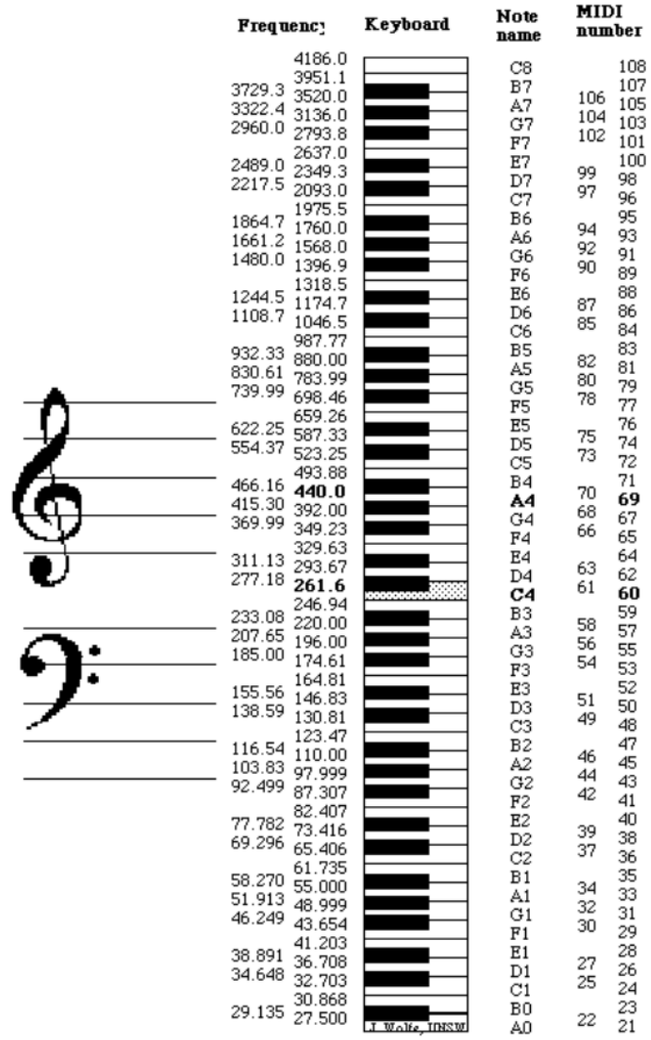
- Sequencer: decide la sequenza di eventi che devono essere eseguiti e i rispettivi parametri
  1. Sequencer sequencer = getSequencer();
  2. sequencer.setTempoInBPM(164.0f);
  3. Sequence seq = **getMidiInputData();**
     - **setMidiEvents();**
       - **addMidiEvent();**
         - setMessage(); **[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( );
1       int instrument = 26;
        sm.setMessage(ShortMessage.PROGRAM_CHANGE,
 2, instrument, 0);
        track.add(new MidiEvent(sm, 0));
2         int channel = 1;
          int velocity = 64;
          int note = 60;
          int tick = 0;
3   addMidiEvent(track, ShortMessage.NOTE_ON, channel, note, velocity, tick);
              tick+=3;
    addMidiEvent(track, ShortMessage.NOTE_OFF, channel, note, 0, tick);
              tick+=1;
    [addMidiEvent…]
```

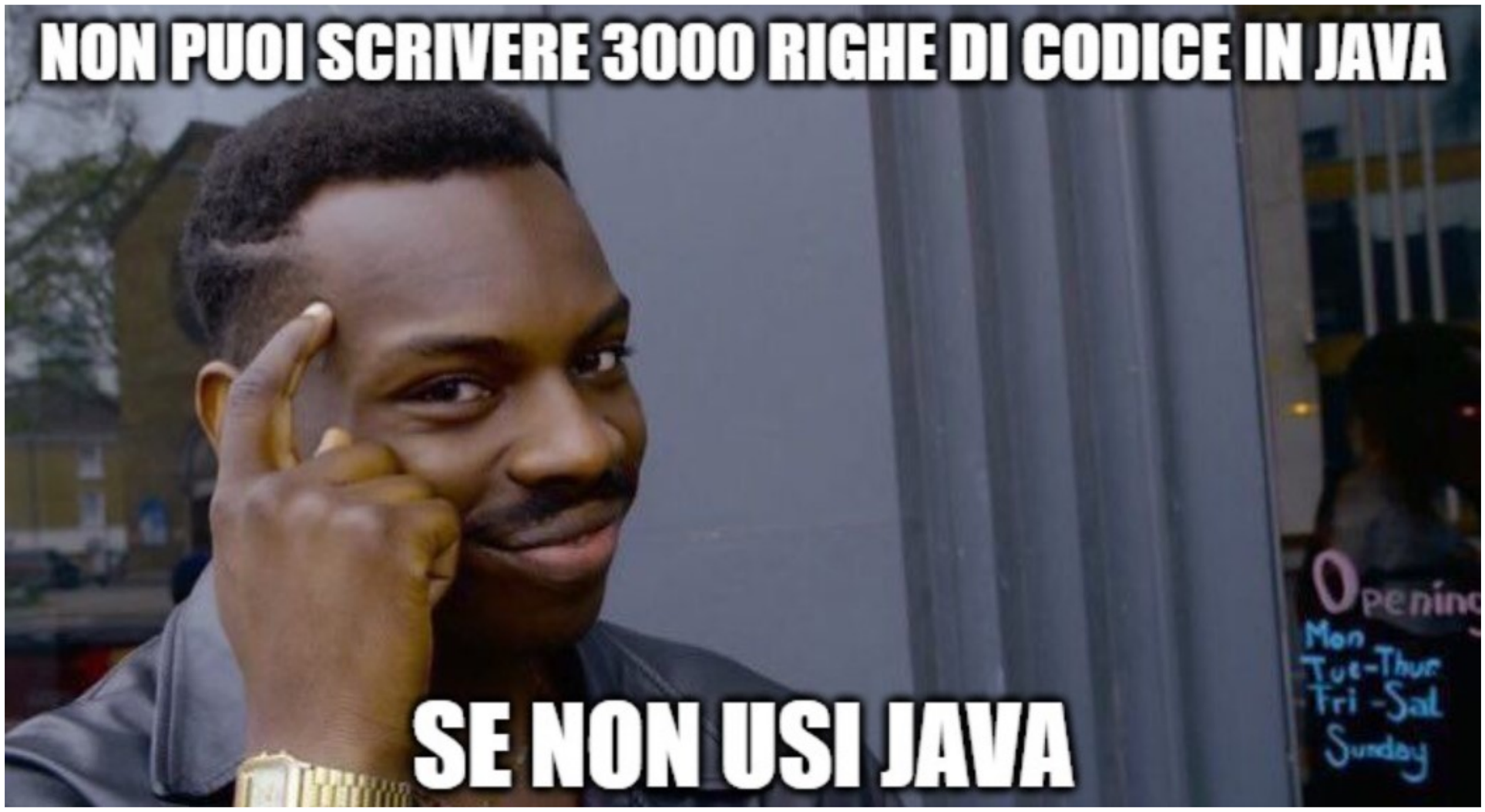
2. Brano in Java (4)

```
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)

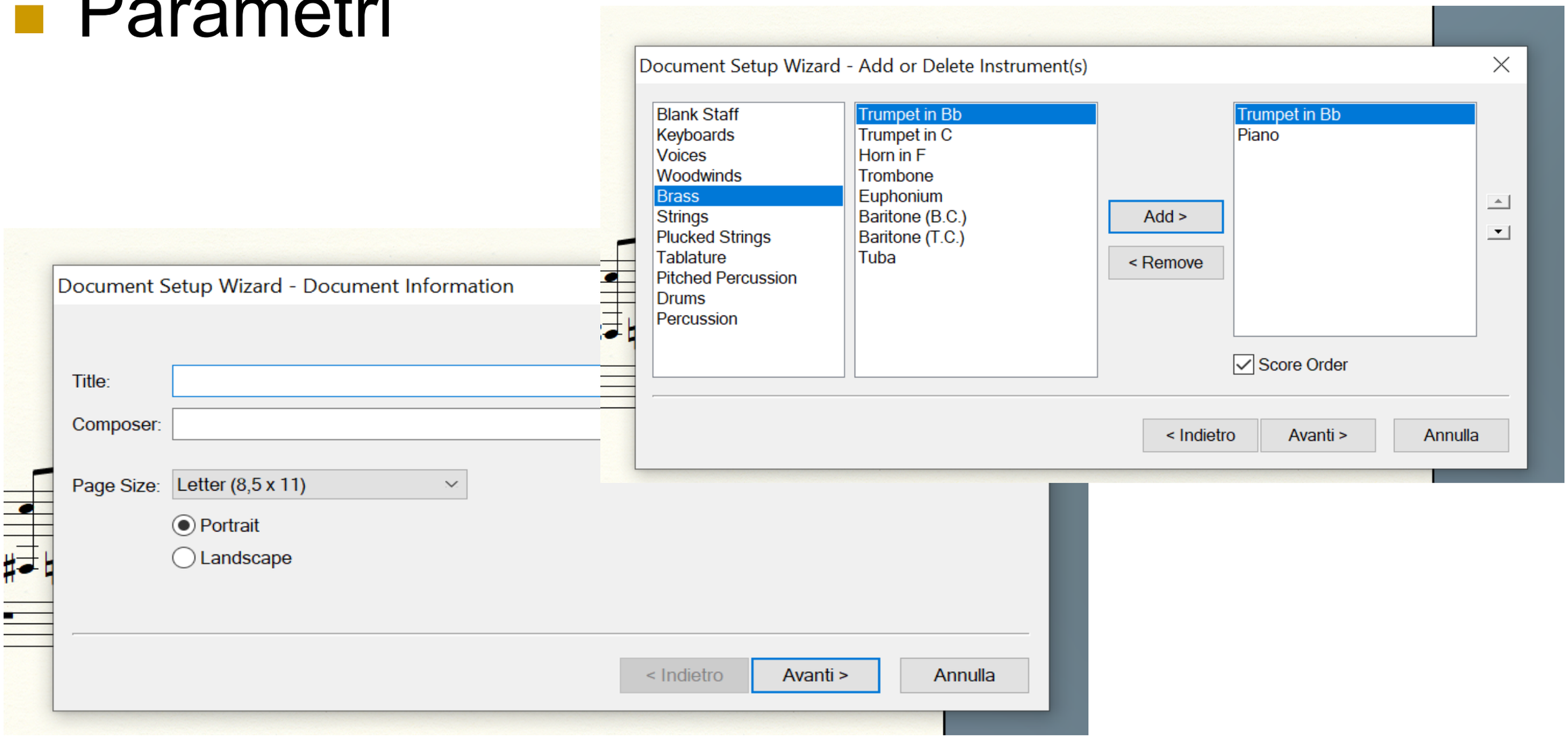
```
private void salvaSuFile(Sequence seq) {
        File f = new File("midifile.mid");
          try {
              MidiSystem.write(seq,1,f);
          } catch (IOException ex) {
              ex.printStackTrace();
           }
        System.out.println("midifile end ");
}
```

2.1 Complessità crescente



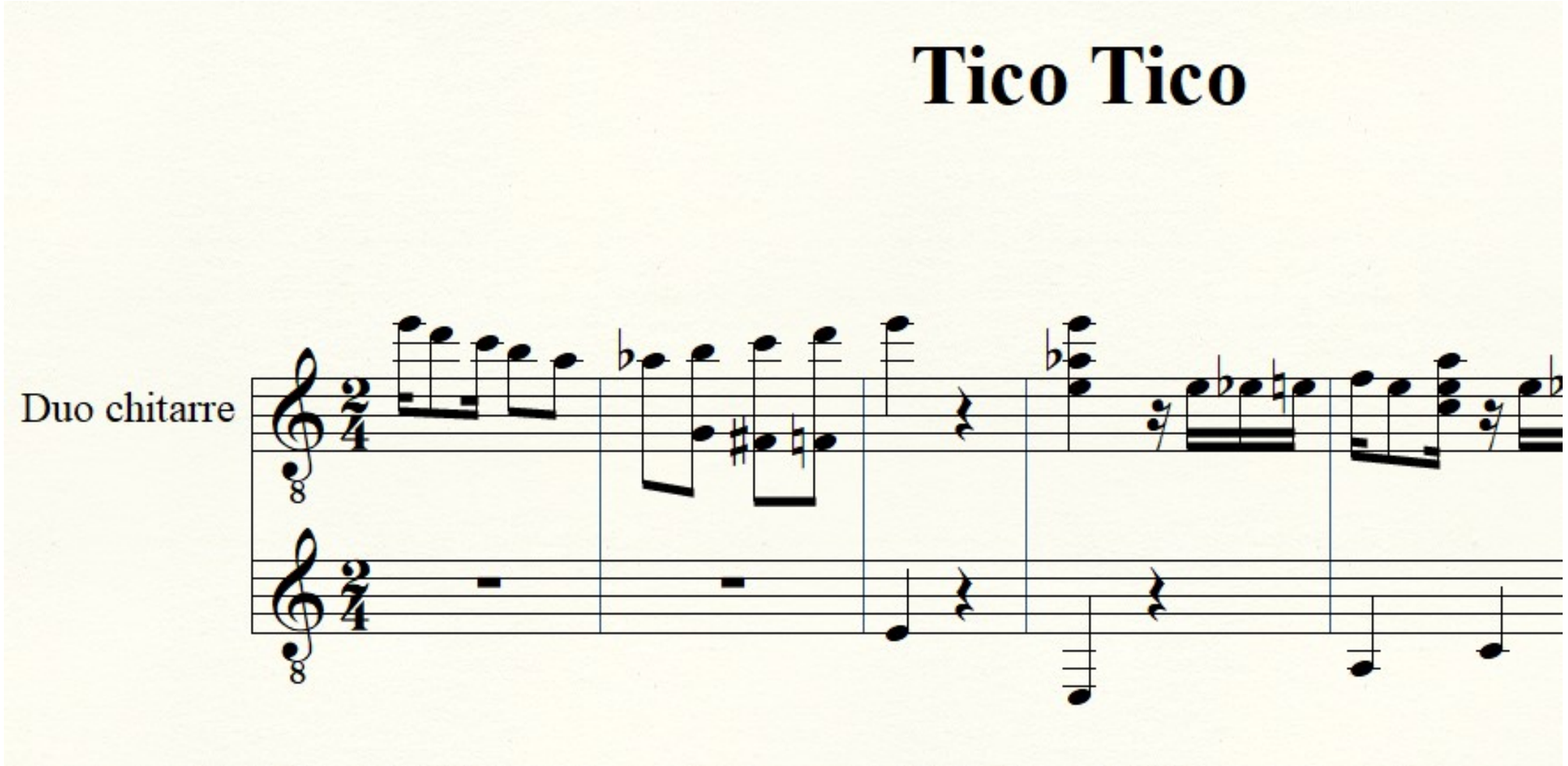
3. Brano in Finale (1)

- Parametri



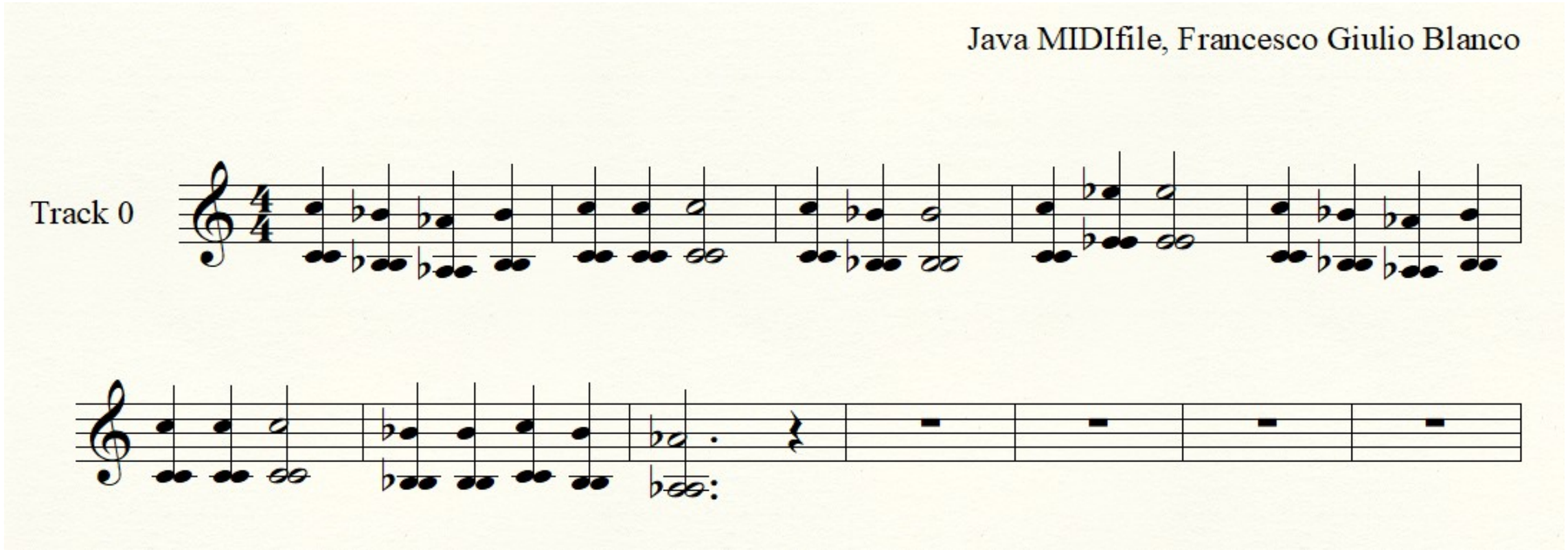
3. Brano in Finale (2)

- Partitura



4. Java → Finale

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



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