



Панов Михаил Александрович

к.э.н., доцент кафедры информационных технологий и статистики «УрГЭУ»

panov79@ya.ru

Создаем BeatBox

Cyber BeatBox

Bass Drum	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Closed Hi-Hat	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Open Hi-Hat	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Acoustic Snare	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Crash Cymbal	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Hand Clap	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
High Tom	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Hi Bongo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Maracas	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Whistle	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Low Conga	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Cowbell	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Vibraslap	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Low-mid Tom	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
High Agogo	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Open Hi Conga	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Start

Stop

Tempo Up

Tempo Down

① Проектируем GUI, который будет иметь 256 снятых флажков (JCheckBox), 16 меток (JLabel) для названий инструментов и четыре кнопки.

② Связываем ActionListener с каждой из четырех кнопок. Слушатели для флажков нам не нужны, потому что мы не будем динамически изменять схему звучания (то есть когда пользователь устанавливает флажки). Вместо этого мы ждем, пока пользователь нажмет кнопку Старт, а затем пробегаем через все 256 флажков, чтобы получить их состояния, и, основываясь на этом, создаем MIDI-дорожку.

③ Устанавливаем систему MIDI (мы делали это раньше), получая доступ к синтезатору, создаем объект Sequencer и дорожку для него. Мы используем новый метод интерфейса Sequencer setLoopCount(), появившийся в Java 5.0. Он позволяет определять желаемое количество циклов последовательности. Мы также будем использовать коэффициент темпа последовательности для настройки уровня темпа и сохранять новый темп от одной итерации цикла к другой.

④ При нажатии пользователем кнопки Старт начинается настоящее действие. Обработчик событий кнопки запускает метод buildTrackAndStart(). В нем мы пробегаем через все 256 флажков (по одному ряду за один раз, один инструмент на все 16 тактов), чтобы получить их состояния, а затем используем эту информацию для создания MIDI-дорожки (с помощью удобного метода makeEvent(), который мы применяли в предыдущей главе). Как только дорожка построена, мы запускаем секвенсор, который будет играть (потому что мы его зацикливаем), пока пользователь не нажмет кнопку Стоп.

```
import java.awt.*;
import javax.swing.*;
import javax.sound.midi.*;
import java.util.*;
import java.awt.event.*;
```

```
public class BeatBox {
```

```
JPanel mainPanel;
ArrayList<JCheckBox> checkboxList;
Sequencer sequencer;
Sequence sequence;
Track track;
JFrame theFrame;
```

Мы храним флажки в массиве ArrayList.

Это названия инструментов в виде строкового массива, предназначенные для создания меток в пользовательском интерфейсе (на каждый ряд).

```
String[] instrumentNames = {"Bass Drum", "Closed Hi-Hat",
    "Open Hi-Hat", "Acoustic Snare", "Crash Cymbal", "Hand Clap",
    "High Tom", "Hi Bongo", "Maracas", "Whistle", "Low Conga",
    "Cowbell", "Vibraslap", "Low-mid Tom", "High Agogo",
    "Open Hi Conga"};
int[] instruments = {35, 42, 46, 38, 49, 39, 50, 60, 70, 72, 64, 56, 58, 47, 67, 63};
```

```
public static void main (String[] args) {
    new BeatBox2().buildGUI();
}
```

```
public void buildGUI() {
    theFrame = new JFrame("Cyber BeatBox");
    theFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    BorderLayout layout = new BorderLayout();
    JPanel background = new JPanel(layout);
    background.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
```

Эти числа представляют собой фактические барабанные клавиши. Канал барабана — это что-то вроде фортепиано, только каждая клавиша на нем — отдельный барабан. Номер 35 — это клавиша для Bass drum, а 42 — Closed Hi-Hat и т.д.

```
checkboxList = new ArrayList<JCheckBox>();
Box buttonBox = new Box(BoxLayout.Y_AXIS);
```

```
JButton start = new JButton("Start");
start.addActionListener(new MyStartListener());
buttonBox.add(start);
```

```
JButton stop = new JButton("Stop");
stop.addActionListener(new MyStopListener());
buttonBox.add(stop);
```

```
JButton upTempo = new JButton("Tempo Up");
upTempo.addActionListener(new MyUpTempoListener());
buttonBox.add(upTempo);
```

```
JButton downTempo = new JButton("Tempo Down");
```

Здесь нет ничего нового — с большей частью кода вы уже знакомы.

```
downTempo.addActionListener(new MyDownTempoListener());  
buttonBox.add(downTempo);
```

```
Box nameBox = new Box(BoxLayout.Y_AXIS);  
for (int i = 0; i < 16; i++) {  
    nameBox.add(new Label(instrumentNames[i]));  
}
```

```
background.add(BorderLayout.EAST, buttonBox);  
background.add(BorderLayout.WEST, nameBox);
```

```
theFrame.getContentPane().add(background);
```

```
GridLayout grid = new GridLayout(16,16);  
grid.setVgap(1);  
grid.setHgap(2);  
mainPanel = new JPanel(grid);  
background.add(BorderLayout.CENTER, mainPanel);
```

```
for (int i = 0; i < 256; i++) {  
    JCheckBox c = new JCheckBox();  
    c.setSelected(false);  
    checkboxList.add(c);  
    mainPanel.add(c);  
} // Конец цикла
```

Создаем флажки, присваиваем им значения false (чтобы они не были установлены), а затем добавляем их в массив ArrayList и на панель.

```
setUpMidi();
```

```
theFrame.setBounds(50,50,300,300);  
theFrame.pack();  
theFrame.setVisible(true);  
} // Закрываем метод
```

Еще код для GUI. Ничего особенного.

```
public void setUpMidi() {  
    try {  
        sequencer = MidiSystem.getSequencer();  
        sequencer.open();  
        sequence = new Sequence(Sequence.PPQ,4);  
        track = sequence.createTrack();  
        sequencer.setTempoInBPM(120);  
  
    } catch (Exception e) {e.printStackTrace();}  
} // Закрываем метод
```

Обычный MIDI-код для получения синтезатора, секвенсора и дорожки. По-прежнему ничего особенного.

Вот здесь все и происходит! Мы преобразуем состояния флажков в MIDI-события и добавляем их на дорожку.

```
public void buildTrackAndStart() {  
    int[] trackList = null;
```

Создаем массив из 16 элементов, чтобы хранить значения для каждого инструмента, на все 16 тактов.

```
sequence.deleteTrack(track);  
track = sequence.createTrack();
```

Издаемся от старой дорожки и создаем новую.

```
for (int i = 0; i < 16; i++) {  
    trackList = new int[16];
```

Делаем это для каждого из 16 рядов (то есть для Bass, Conga и т.д.).

```
int key = instruments[i];
```

Задаем клавишу, которая представляет инструмент (Bass, Hi-Hat и т.д.). Массив содержит MIDI-числа для каждого инструмента.

```
for (int j = 0; j < 16; j++) {
```

Делаем это для каждого такта текущего ряда.

```
    JCheckBox jc = (JCheckBox) checkboxList.get(j + (16*i));  
    if (jc.isSelected()) {  
        trackList[j] = key;  
    } else {  
        trackList[j] = 0;  
    }  
} // Закрываем внутренний цикл
```

Установлен ли флажок на этом такте? Если да, то помещаем значение клавиши в текущую ячейку массива (ячейку, которая представляет такт). Если нет, то инструмент не должен играть в этом такте, поэтому присвоим ему 0.

```
makeTracks(trackList);  
track.add(makeEvent(176,1,127,0,16));  
} // Закрываем внешний
```

Для этого инструмента и для всех 16 тактов создаем события и добавляем их на дорожку.

```
track.add(makeEvent(192,9,1,0,15));  
try {
```

Мы всегда должны быть уверены, что событие на такте 16 существует (они идут от 0 до 15). Иначе BeatBox может не пройти все 16 тактов, перед тем как заново начнет последовательность.

```
    sequencer.setSequence(sequence);  
    sequencer.setLoopCount(sequencer.LOOP_CONTINUOUSLY);  
    sequencer.start();  
    sequencer.setTempoInBPM(120);  
} catch (Exception e) {e.printStackTrace();}  
} // Закрываем метод buildTrackAndStart
```

Позволяет задать количество повторений цикла или, как в этом случае, непрерывный цикл.

Теперь мы проигрываем мелодию!

```
public class MyStartListener implements ActionListener {  
    public void actionPerformed(ActionEvent a) {  
        buildTrackAndStart();  
    }  
} // Закрываем внутренний класс
```

Первый из внутренних классов — слушателей для кнопок.

```
public class MyStopListener implements ActionListener {
    public void actionPerformed(ActionEvent a) {
        sequencer.stop();
    }
} // Закрываем внутренний класс
```

```
public class MyUpTempoListener implements ActionListener {
    public void actionPerformed(ActionEvent a) {
        float tempoFactor = sequencer.getTempoFactor();
        sequencer.setTempoFactor((float) (tempoFactor * 1.03));
    }
} // Закрываем внутренний класс
```

```
public class MyDownTempoListener implements ActionListener {
    public void actionPerformed(ActionEvent a) {
        float tempoFactor = sequencer.getTempoFactor();
        sequencer.setTempoFactor((float) (tempoFactor * .97));
    }
} // Закрываем внутренний класс
```

Другие внутренние классы — слушатели для кнопок.

← Коэффициент темпа определяет темп синтезатора. По умолчанию он равен 1.0, поэтому щелчком кнопкой мыши можно изменить его на +/- 3 %.

```
public void makeTracks(int[] list) {
    for (int i = 0; i < 16; i++) {
        int key = list[i];

        if (key != 0) {
            track.add(makeEvent(144, 9, key, 100, i));
            track.add(makeEvent(128, 9, key, 100, i+1));
        }
    }
}
```

Метод создает события для одного инструмента за каждый проход цикла для всех 16 тактов. Можно получить int[] для Bass drum, и каждый элемент массива будет содержать либо клавишу этого инструмента, либо ноль. Если это ноль, то инструмент не должен играть на текущем такте. Иначе нужно создать событие и добавить его в дорожку.

← Создаем события включения и выключения и добавляем их в дорожку.

```
public MidiEvent makeEvent(int comd, int chan, int one, int two, int tick) {
    MidiEvent event = null;
    try {
        ShortMessage a = new ShortMessage();
        a.setMessage(comd, chan, one, two);
        event = new MidiEvent(a, tick);

    } catch (Exception e) {e.printStackTrace(); }

    return event;
}

} // Закрываем класс
```

← Это полезный метод из предыдущей главы Кухни кода.

