Exercise – MusicTest1.java

```
1  package Exceptions;
2
3  // musical Instrument Digital Interface
6
7  import javax.sound.midi.*;
8
9  // first get a sequencer object - takes in the midi data
10 // and sends it to the right place -- it plays the music
11
12 public class MusicTest1 {
13     public void play() {
14
15         // getSequencer() check the Oracle docs to determine what
16         // exceptions it throws and use it in the catch
17         //
18         // here we are handling the exception
19
20         try {
21             Sequencer sequencer = MidiSystem.getSequencer();
22             System.out.println("Successfully got a sequencer");
23         } catch (MidiUnavailableException ex) {
24             System.out.println("Didn't get one");
25         }
26     } // close play
27
28     public static void main (String [] args){
29         MusicTest1 mt = new MusicTest1();
30         mt.play();
31     } // close main
32 } // close class
```

```
Problems  Javadoc  Declaration  Console
<terminated> MusicTest1 [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (Nov 25, 2014, 10:16:14 PM)
Successfully got a sequencer
```

Exercise - Custom Exceptions

- Sometimes we need to create custom exceptions based on the application requirements
- We create our own exceptions by extending the Exception class
- This is demonstrated in the following example:
- Code the following and test as Java application when input is valid and when the exception is forced
  - Uncomment line 6 then run
  - Comment line 6, uncomment line 7 then run

```java
package Exceptions;

public class MyOwnException {
    public static void main(String[] a){
        try{
            //MyOwnException.myTest("this is a test"); // test not null string
            //MyOwnException.myTest(null);   // test null string w/ exception

        } catch(MyAppException mae){
            System.out.println("Inside catch block: "+mae.getMessage());
        }
    } // end of main

    static void myTest(String str) throws MyAppException{
        if(str == null){
            throw new MyAppException("String val is null");
        }else {
            System.out.println("All's good!");
        }
    } // end of myTest
} // end of class MyOwnException

class MyAppException extends Exception {

    private String message = null;
    public MyAppException() {
        super();
    }

    public MyAppException(String message) {
        super(message);
        this.message = message;
    }

    public MyAppException(Throwable cause) {
        super(cause);
    }

    @Override
    public String toString() {
        return message;
    }

    @Override
    public String getMessage() {
        return message;
    }
} // end of class MyAppException
```

```
<terminated> MyOwnException [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (Nov 25, 2014, 10:34:47 PM)
All's good!
```

Expand MusicTest1 to play something…

- In this exercise we expand the original MusicTest1.java to actually play

```
1  package Exceptions;
2
3  import javax.sound.midi.*;
4
5  public class MiniMiniMusicApp {
6      public static void main(String [] args){
7          MiniMiniMusicApp mini = new MiniMiniMusicApp();
8          mini.play();
9      } // close main
10
11     public void play(){
12         try {
13             Sequencer player = MidiSystem.getSequencer();
14             player.open();
15
16             Sequence seq = new Sequence(Sequence.PPQ, 4);
17             Track track = seq.createTrack();
18
19             //  music played as messages - set the instrument,
20             //   set the message (music note), add it to the track, set the seq
21             //   start the player
22
23
24             //-- message 192 says change the instrument
25             ShortMessage first = new ShortMessage();
26             first.setMessage(192, 1, 102, 0); //- default is piano, 102 is sax
27             MidiEvent changeInstrument = new MidiEvent(first,1);
28             track.add(changeInstrument);
29
30             // 144 = message type
31             // 1 = channel - musician 1
32             // 44 = note to play (0 - 127 low to high)
33             // 100 = velocity (how hard and fast to press the key
34
35             ShortMessage a = new ShortMessage();
36             a.setMessage(144, 1, 44, 100);
37             MidiEvent noteOn = new MidiEvent(a, 1); // duration
38             track.add(noteOn);     // start playing
39
40             ShortMessage b = new ShortMessage();
41             b.setMessage(128, 1, 44, 100);
42             MidiEvent noteOff = new MidiEvent(b, 16);
43             track.add(noteOff);    // stop playing
44
45             player.setSequence(seq);
46             player.start();
47         } catch (Exception ex) {
48             ex.printStackTrace();
49         }
50     } // end play
51 } // class close
```

Basically building the tracks of a cd

Independent Exercise – Expand MiniMiniMusicApp.java

- Use variables for instrument and note replacing the following

  mini.play(); on line 8

  With

  mini.play(instrument, note);

  Both variables should be declared as integers and have valid values of 0 – 127

  Test with the following values

  1 & 120 – Acoustic Grand Piano and high note of 120

  11 & 90 – Music Box and lower note

  14 & 80 – Xylophone and lower note

Solution – MiniMusic2.java

```java
package Exceptions;

import javax.sound.midi.*;

public class MiniMusic2 {
    public static void main (String [] args){
        MiniMusic2 mini = new MiniMusic2();

        int instrument = 1;  // Acoustic Grand Piano
        int note = 120;
        mini.play(instrument, note);

        int instrument2 = 11; // Music Box
        int note2 = 90;
        mini.play(instrument2, note2);

        int instrument3 = 14; // xylophone
        int note3 = 80;
        mini.play(instrument3, note3);

    }// end of main

    public void play(int instrument, int note){
        try {
            Sequencer player = MidiSystem.getSequencer();
            player.open();
            Sequence seq = new Sequence(Sequence.PPQ, 4);
            Track track = seq.createTrack();

            MidiEvent event = null;

            ShortMessage first = new ShortMessage();
            first.setMessage(192, 1, instrument, 0);
            MidiEvent changeInstrument = new MidiEvent(first,1);
            track.add(changeInstrument);

            ShortMessage a = new ShortMessage();
            a.setMessage(144, 1, note, 100);
            MidiEvent noteOn = new MidiEvent(a, 1);
            track.add(noteOn);

            ShortMessage b = new ShortMessage();
            b.setMessage(128, 1, note, 100);
            MidiEvent noteOff = new MidiEvent(b, 16);
            track.add(noteOff);

            player.setSequence(seq);
            player.start();

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

}  // end of MiniMusic2
```