13. 예외 처리

음악 재생 프로그램

예외 처리 방법

try/catch 블록

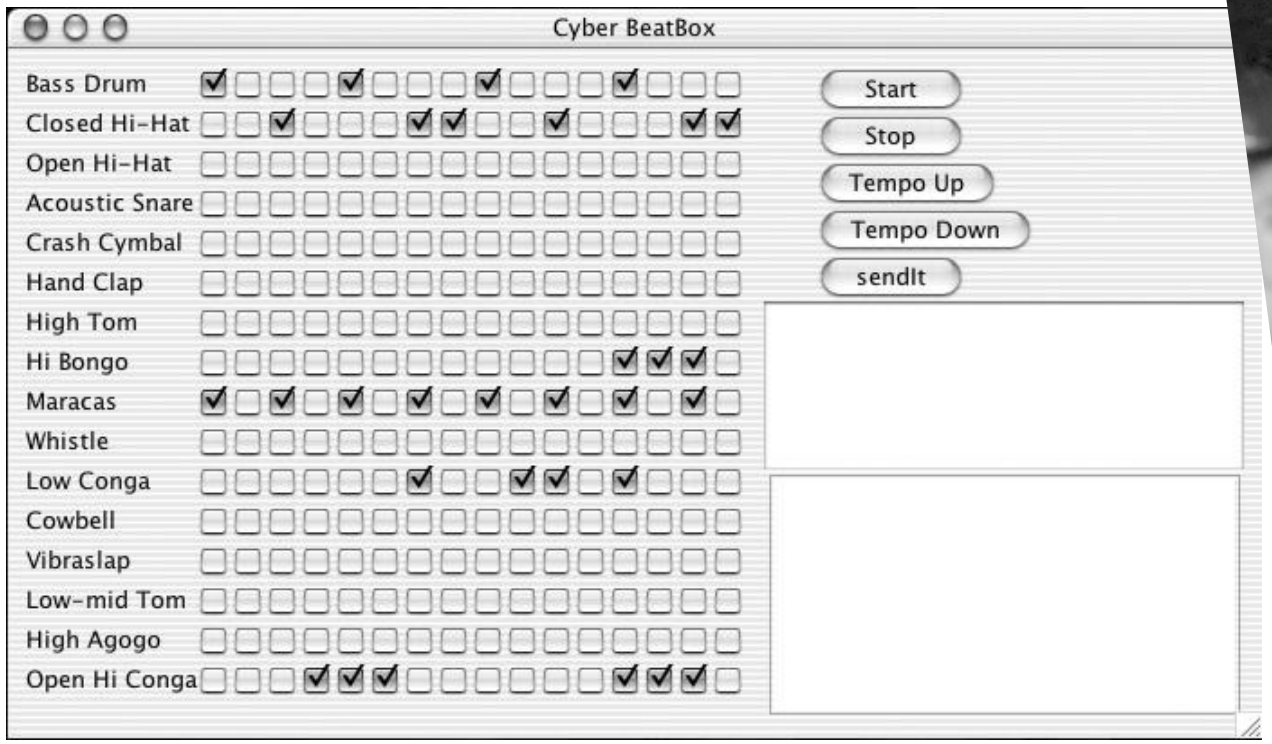
예외 선언 방법

위험한 행동

- 예상치 못한 상황
  - 파일이 없는 경우
  - 서버가 다운되는 경우
  - 장치를 사용할 수 없는 경우

- 이런 예외적인 상황을 처리하기 위한 방법이 필요합니다.
  - 자바의 예외 처리 메커니즘
  - try/catch[/finally] 블록
  - 예외 선언

음악 재생 프로그램


Cyber BeatBox
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
Start
Stop
Tempo Up
Tempo Down
sendIt

JavaSound API

- JavaSound API
  - MIDI (javax.sound.midi 패키지)
    - 악기 디지털 인터페이스(Musical Instruments Digital Interface)
  - Sampled

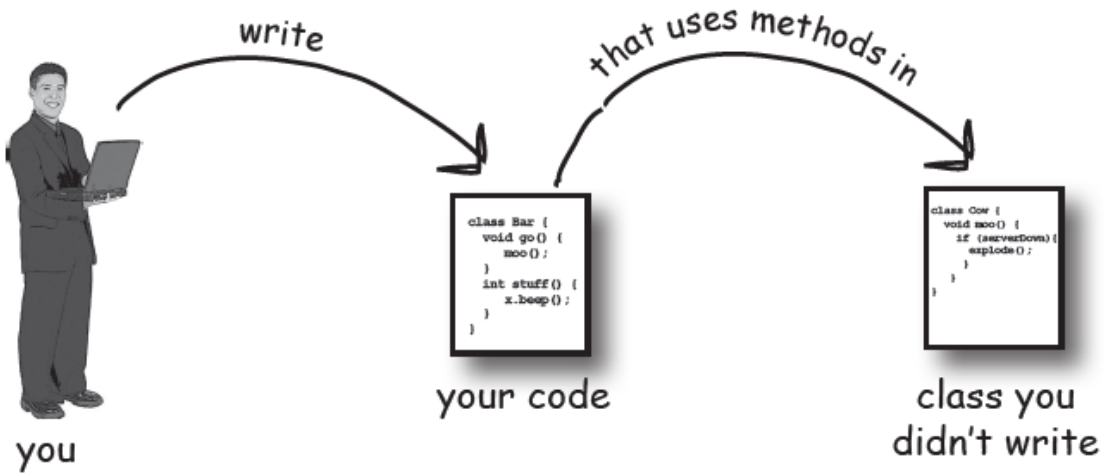
Sequencer

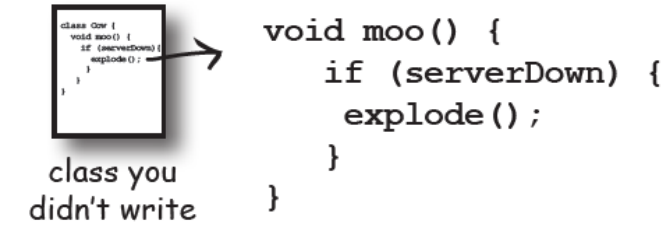
```
import javax.sound.midi.*;

public class MusicTest1 {
  public void play() {
    Sequencer sequencer = MidiSystem.getSequencer();
    System.out.println("We got a sequencer");
  }
  public static void main(String[] args) {
    MusicTest1 mt = new MusicTest1();
    mt.play();
  }
}
```

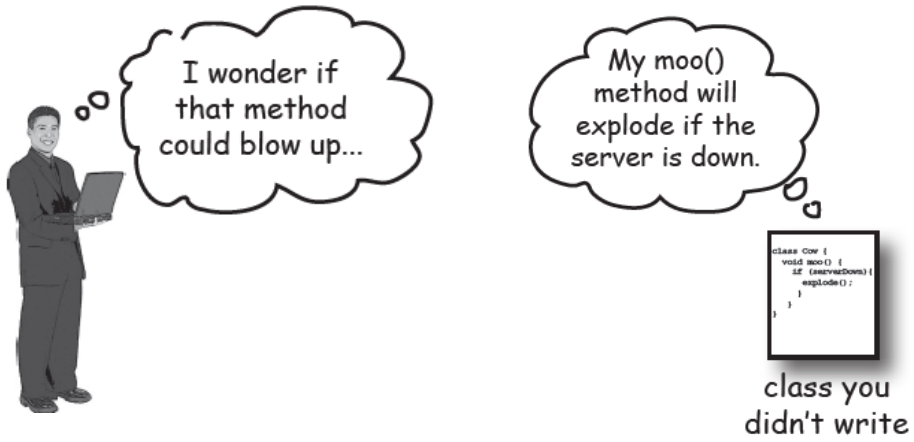
```
$ javac MusicTest1.java
MusicTest1.java:5: unreported exception javax.sound.midi.MidiUnavailableException; must be caught
or declared to be thrown
    Sequencer sequencer = MidiSystem.getSequencer();
                                     ^
1 error
```

p.466

위험 요소가 있는 메소드







p.467

예외 처리 메커니즘

- 예외(exception)
  - 프로그램 실행 중에 생길 수 있는 특이한 상황
- 예외 처리(exception handling) 메커니즘
  - 오류 처리 코드를 한 군데로 모아 놓을 수 있기 때문에 코드가 깔끔해지고 효율적이 됩니다.

javax.sound.midi

Class MidiSystem

java.lang.Object
    javax.sound.midi.MidiSystem

```
public class MidiSystem
extends Object
```

getSequencer

```
public static Sequencer getSequencer()
                          throws MidiUnavailableException
```

throws 절(clause)가 있으면 예외 발생됨을 알 수 있다.

p.468

try/catch 블록

- 예외를 처리할 것임을 알려주기 위한 용도로 쓰임

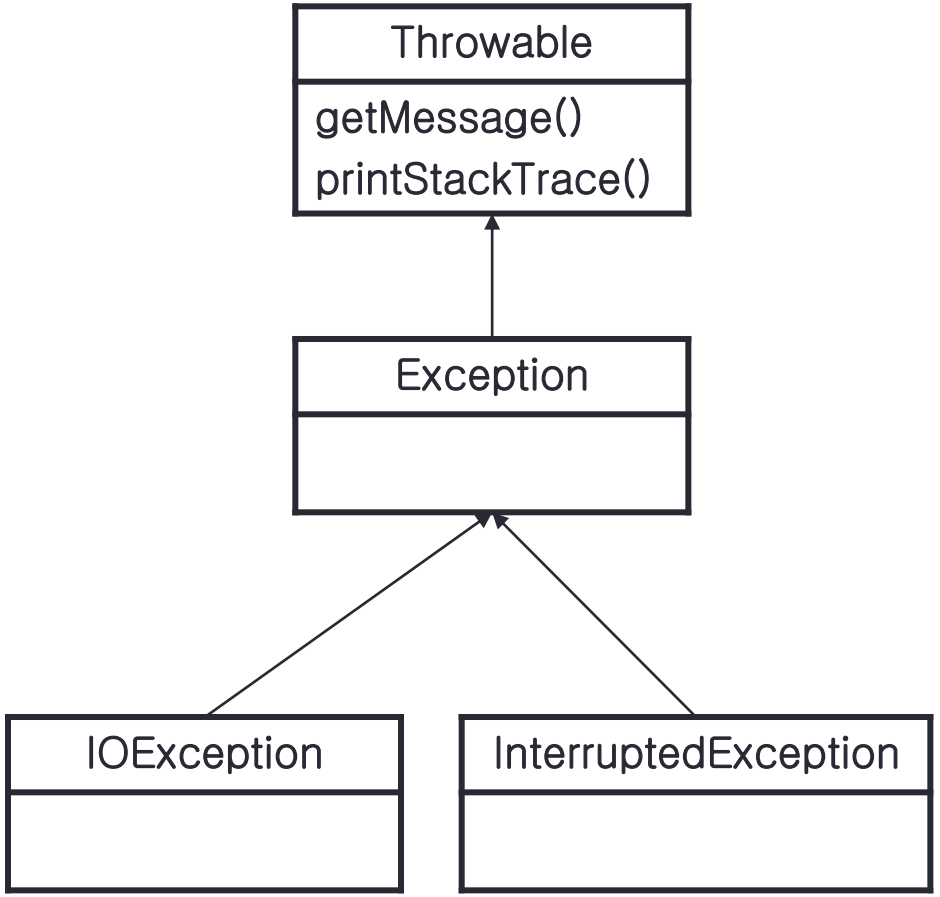
```
import javax.sound.midi.*;

public class MusicTest1 {
  public void play() {
    try {
      Sequencer sequencer = Midisystem.getSequencer();
      System.out.println("Successfully got a sequencer");
    } catch (MidiUnavailableException ex) {
      System.out.println("Bummer");
    }
}

  public static void main(String[] args) {
    MusicTest1 mt = new MusicTest1();
    mt.play();
  }
}
```

public void play() throws ~Exception {

p.469

Exception 클래스

```
try {
  // 위험한 일 처리
} catch (Exception ex) {
  // 문제 해결
}
```



p.470

예외 던지기와 잡기

- 예외를 던지는 코드

```
public void takeRisk() throws BadException {
  if (abandonAllHope) {
    throw new BadException();
  }
}
```

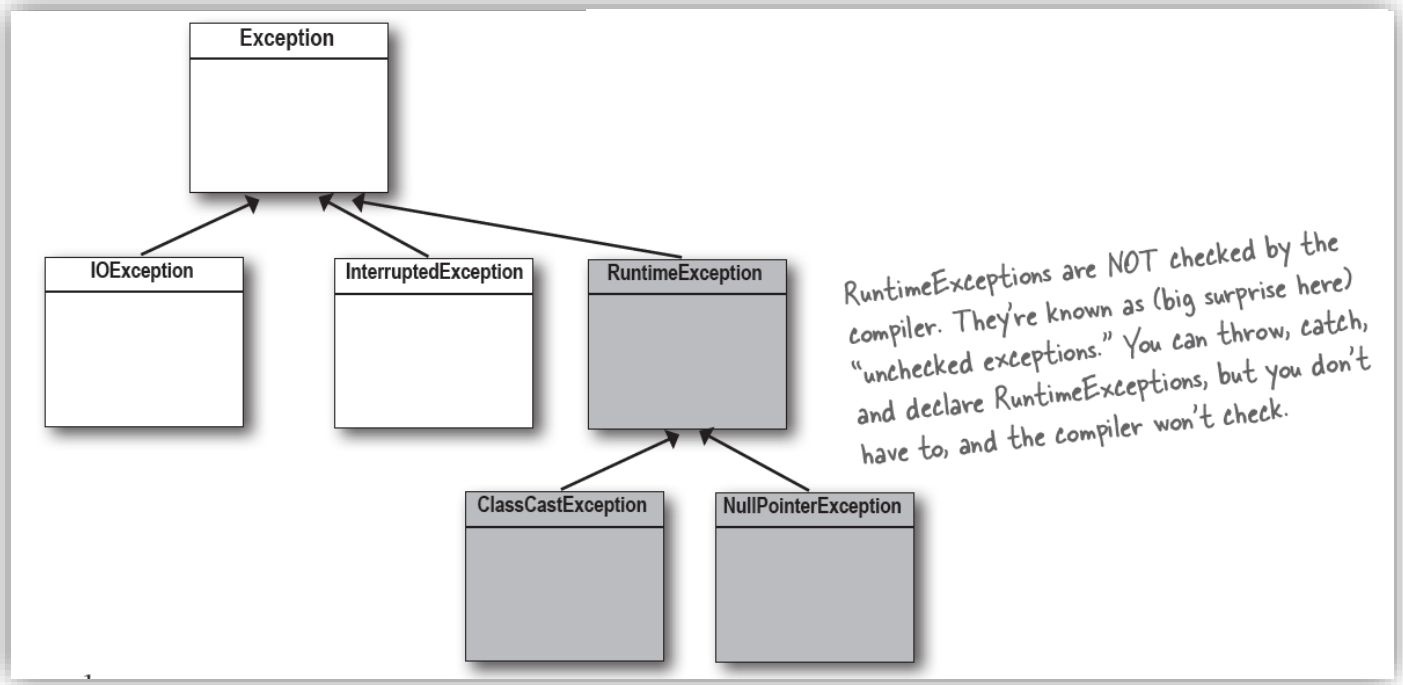
- 위험한 메소드를 호출하는 코드

```
public void crossFingers() {
  try {
    anObject.takeRisk();
  } catch (BadException ex) {
    System.out.println("Aaargh!");
    ex.printStackTrace();
  }
}
```

p.471

확인 예외

- 확인 예외(checked exception)
  - RuntimeException을 제외한 모든 예외는 컴파일러에서 확인
  - 코드에서 예외를 던진다면 반드시 메소드를 선언하는 부분에서 throws 키워드를 써야 함
  - 예외를 던지는 메소드를 호출하면 try/catch 블록으로 그 부분을 감싸거나 그 메소드에서도 예외를 선언해야 함



p.472

바보 같은 질문은 없습니다.

- NullPointerException, DivideByZero 같은 예외에 대해서는 왜 try/catch 블록을 사용하지 않나요?
  - RuntimeException 및 그 하위클래스에 속하는 예외는 컴파일러에서 잡아내지 않습니다.
  - 런타임 예외는 실행 중에 어떤 조건에 문제가 생기는 경우보다는 코드의 논리에 예측/예방할 수 없는 방식으로 문제가 생기는 경우에 발생합니다.
  - try/catch 블록은 예외적인 상황을 처리하기 위한 것이지 코드의 문제점을 보완하기 위한 것은 아닙니다.

try/catch 블록의 흐름 제어

```
try {
   Foo f = x.doRiskyThing();
   int b = f.getNum();
} catch (Exception ex) {
   System.out.println("failed");
}
System.out.println("We made it!");
```

try 블록 성공

```
$ java Tester
We made it!
```

try 블록 실패

```
$ java Tester
failed
We made it!
```

무조건 실행할 내용

- 예외 발생 여부와 상관없이 무조건 실행할 코드는 finally 블록에...
- try 또는 catch 블록에 return 문이 있어도 finally는 실행됨.
  제어 흐름이 finally 블록으로 넘어갔다가 return됨.

```
try {
    turnOvenOn();
    x.bake();
} catch (BakingException ex) {
    ex.printStackTrace();
} finally {
    turnOvenOff();
}
```

```
try {
    turnOvenOn();
    x.bake();
    turnOvenOff();
} catch (BakingException ex) {
    ex.printStackTrace();
    turnOvenOff();
}
```

p.475

더 간단한 방법: try-with-resources 문장

- 참고 URL: https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html
- **try (자원 열기) { ... }** 형태로 사용 (since Java SE 7)
  - 자원 닫기 필요 없음.

```
static String readFirstLineFromFile(String path) throws IOException {
    try (FileReader fr = new FileReader(path);
         BufferedReader br = new BufferedReader(fr)) {
        return br.readLine();
    }
}
```

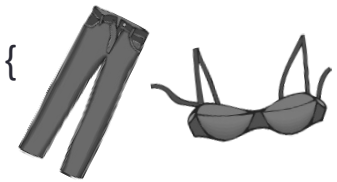
- Java SE 7 이전의 코드

```
static String readFirstLineFromFileWithFinallyBlock(String path) throws IOException {

    FileReader fr = new FileReader(path);
    BufferedReader br = new BufferedReader(fr);
    try {
        return br.readLine();
    } finally {
        br.close();
        fr.close();
    }
}
```

두 개 이상의 예외

- 예외를 여러 개 던진다면 모든 확인 예외를 잡아야 함.
- try 블록 아래 catch 블록을 필요한 예외 개수만큼 넣으면 됨.
- catch 블록에 있는 예외 간 상속 관계 있는 경우 catch 블록의 순서 중요

```
public class Laundry {
  public void doLaundry() throws PantsException, LingerieException {
    // 두 가지 예외를 던질 수 있는 코드
  }
}

public class Foo {
  public void go() {
    Laundry laundry = new Laundry();
    try {
      laundry.doLaundry();
    } catch (PantsException pex) {
    } catch (LingerieException lex) {
    }
  }
}
```



p.477

예외와 다형성: 상위 예외로 하위 예외 처리

```
public void doLaundry throws ClothingException { … }
```

```
try {
   laundry.doLaundry();
} catch (ClothingException cex) {
  // 복구 코드
}
```

```
try {
   laundry.doLaundry();
} catch (ShirtException sex) {
   // 복구 코드
}
```

예외 처리 순서: 작은 것부터 큰 것으로

```
try {
    laundry.doLaundry();
} catch(TeeShirtException tex) {

} catch(LingerieException lex) {

} catch(ClothingException cex) {

}
```

여러 개의 catch 블록을 쓸 때는 상/하위 클래스 관계를 잘 따져봐야 합니다.

Don't do this!

```
try {
  laundry.doLaundry();
}  catch(ClothingException cex) {
  // recovery from ClothingException
}  catch(LingerieException lex) {
  // recovery from LingerieException
}  catch(ShirtException shex) {
  // recovery from ShirtException
}
```

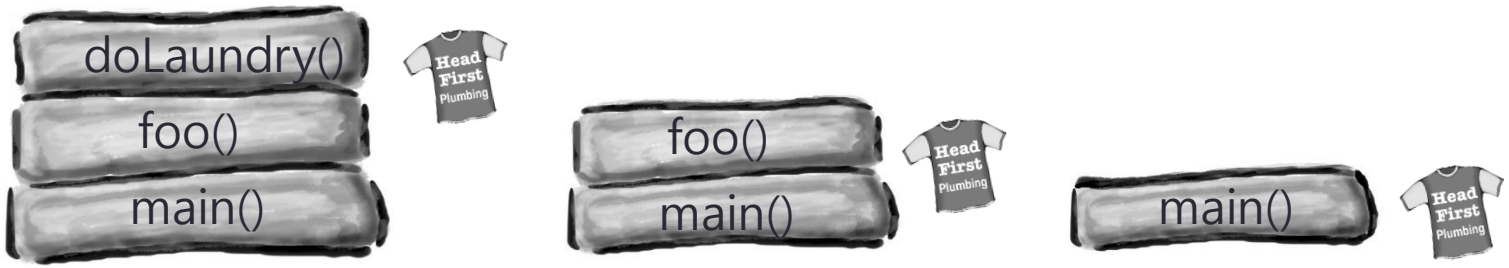
예외 선언

- try/catch 블록을 쓰는 대신 메소드에서 예외를 선언함으로써 예외 처리를 회피하는 방법도 있습니다.

```
public void foo() throws ClothingException { // 교재: ReallyBadException
    // try/catch 블록 없이 위험한 메소드 호출
    laundry.doLaundry();
}
```

예외 선언

```
public class Washer {
  Laundry laundry = new Laundry();
  public void foo() throws ClothingException {
      laundry.doLaundry();
  }
  public static void main(String[] args) throws ClothingException {
      Washer a = new Washer();
      a.foo();
  }
}
```



p.484

실습: Washer.java

```java
package cse.oop2.ch13_3;


public class Washer {

    Laundry laundry = new Laundry();

    public void foo() throws ClothingException {
        laundry.doLaundry();
    }

    public static void main(String[] args) throws ClothingException {
        Washer w = new Washer();
        w.foo();
    }

}
```

Q. foo() 메서드에서 예외 처리를 하려면?

```
20  class Laundry {
21
22      public void doLaundry() throws ClothingException {
23          throw new ClothingException();
24      }
25
26  }
27
28
29  class ClothingException extends Exception {
30
31      public ClothingException() {
32          super("ClothingException");
33      }
34
35  }
```

어느 클래스의 메서드에서도
특정 예외를 처리하지 않으면 JVM에서
Exception.printStackTrace()
사용하여 처리함.

```
PS D:₩User₩jongmin₩강의₩강의노트₩2024-2₩2-JAVA₩SW₩LJM추가코드> java Washer.java
Exception in thread "main" cse.oop2.ch13_3.ClothingException: ClothingException
        at cse.oop2.ch13_3.Laundry.doLaundry(Washer.java:23)
        at cse.oop2.ch13_3.Washer.foo(Washer.java:9)
        at cse.oop2.ch13_3.Washer.main(Washer.java:14)
PS D:₩User₩jongmin₩강의₩강의노트₩2024-2₩2-JAVA₩SW₩LJM추가코드>
```

다시 음악 코드로...

```
import javax.sound.midi.*;

public class MusicTest1 {
  public void play() {
    try {
      Sequencer sequencer = MidiSystem.getSequencer();
      System.out.println("Successfully got a sequencer");
    } catch(MidiUnavailableException ex) {
      System.out.println("Bummer");
    }
  }
  public static void main(String[] args) {
    MusicTest1 mt = new MusicTest1();
    mt.play();
  }
}
```

p.486

예외와 관련된 규칙

1. try 없이 catch나 finally만 쓸 수는 없음

```
void go() {
  Foo f = new Foo();
  f.foof();
  catch(FooException ex) { }
}
```

2. try와 catch 사이에 코드를 집어넣을 수 없음

```
try {
  x.doStuff();
}
int y = 43;
} catch(Exception ex) { }
```

p.486

예외와 관련된 규칙

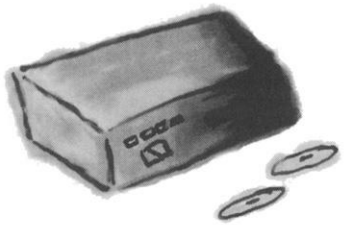
3. try 뒤에는 반드시 catch나 finally가 있어야 함

```
try {
  x.doStuff();
} finally { }
```

4. try 뒤에 finally만 있으면 예외 선언해야 함
   → 예외 처리하는 catch문 없음

```
void go() throws FooException {
  try {
    x.doStuff();
  } finally { }
}
```

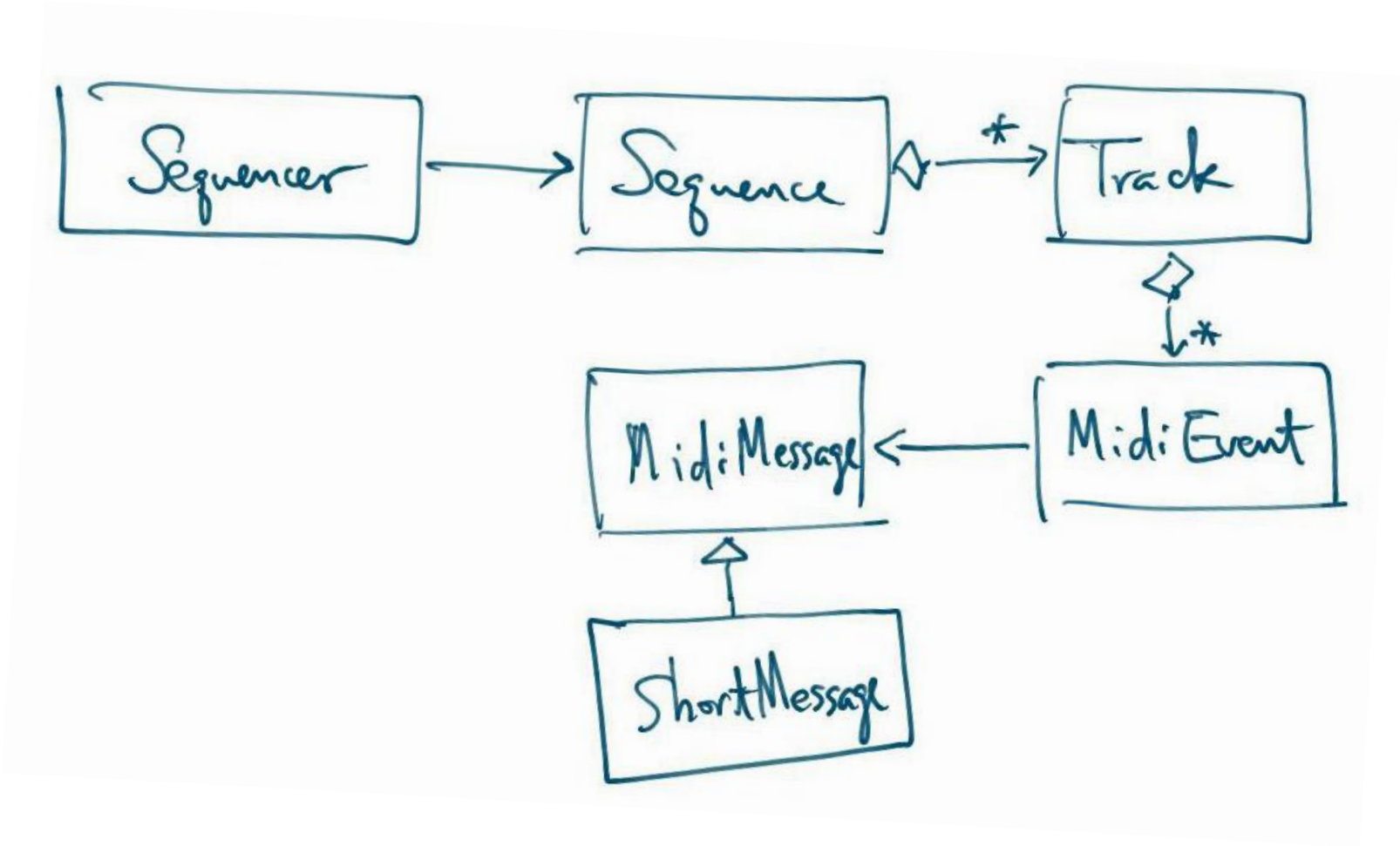
JavaSound API



- 음악을 재생하는 장치
  - Sequencer – CD 플레이어
  - MIDI 시퀀스를 재생할 수 있는 HW/SW 장치
- 재생할 음악
  - Sequence – 곡이 들어 있는 음악 CD
  - Sequencer가 재생할 수 있는 음악 정보를 가지고 있는 자료 구조
- 실제 정보가 들어있는 부분
  - Track – 연주할 악보
  - 표준 MIDI 파일에 다른 트랙과 같이 저장될 수 있는 MIDI 이벤트의 독립적인 스트림
- 실제 음악 정보
  - MidiEvent - Sequencer가 이해할 수 있는 메시지
  - MIDI 메시지와 틱(tick) 형식의 타임 스탬프

p.488

MIDI 관련 클래스 다이어그램



JavaSound API

376 페이지에 있는 코드를 실행 시켜봅시다.

1. Sequencer 만들고 열기

```
Sequencer player = MidiSystem.getSequencer();  // Singleton pattern
```

2. 새로운 Sequence 만들기

```
Sequence seq = new Sequence(timing, 4);
```

3. Sequence에서 새로운 Track 가져오기

```
Track t = seq.createTrack();
```

4. Track에 MidiEvent를 채우고 Sequence를 Sequencer에 넘기기

```
t.add(myMidiEvent1);
player.setSequence(seq);
```

5. Sequencer에 대해 play() 메소드 호출

```
player.start();
```

p.489

MiniMiniMusicApp.java

```
import javax.sound.midi.*;

public class MiniMiniMusicApp {

     public static void main(String[] args) {
        MiniMiniMusicApp mini = new MiniMiniMusicApp();
        mini.play();
     } // close main

    public void play() {

      try {

        Sequencer player = MidiSystem.getSequencer();
        player.open();

        Sequence seq = new Sequence(Sequence.PPQ, 4);
```

Don't forget to import the midi package

get a Sequencer and open it (so we can use it... a Sequencer doesn't come already open)

Don't worry about the arguments to the Sequence constructor. Just copy these (think of 'em as Ready-bake arguments).

참고. tick 유형:

PPQ(Pulse Per Quarter)

SMPTE(Society of Motion Pictures and Television Engineers, 표준)

p.490

```
3  Track track = seq.createTrack();

4  ShortMessage a = new ShortMessage();
   a.setMessage(144, 1, 44, 100);
   MidiEvent noteOn = new MidiEvent(a, 1);
   track.add(noteOn);

   ShortMessage b = new ShortMessage();
   b.setMessage(128, 1, 44, 100);
   MidiEvent noteOff = new MidiEvent(b, 16);
   track.add(noteOff);

   player.setSequence(seq);

   player.start();

    } catch (Exception ex) {
        ex.printStackTrace();
    }
  } // close play
} // close class
```

Ask the Sequence for a Track. Remember, the Track lives in the Sequence, and the MIDI data lives in the Track.

Put some MidiEvents into the Track. This part is mostly Ready-bake code. The only thing you'll have to care about are the arguments to the setMessage() method, and the arguments to the MidiEvent constructor. We'll look at those arguments on the next page.

Give the Sequence to the Sequencer (like putting the CD in the CD player)

Start() the Sequencer (like pushing PLAY)

MidiEvent

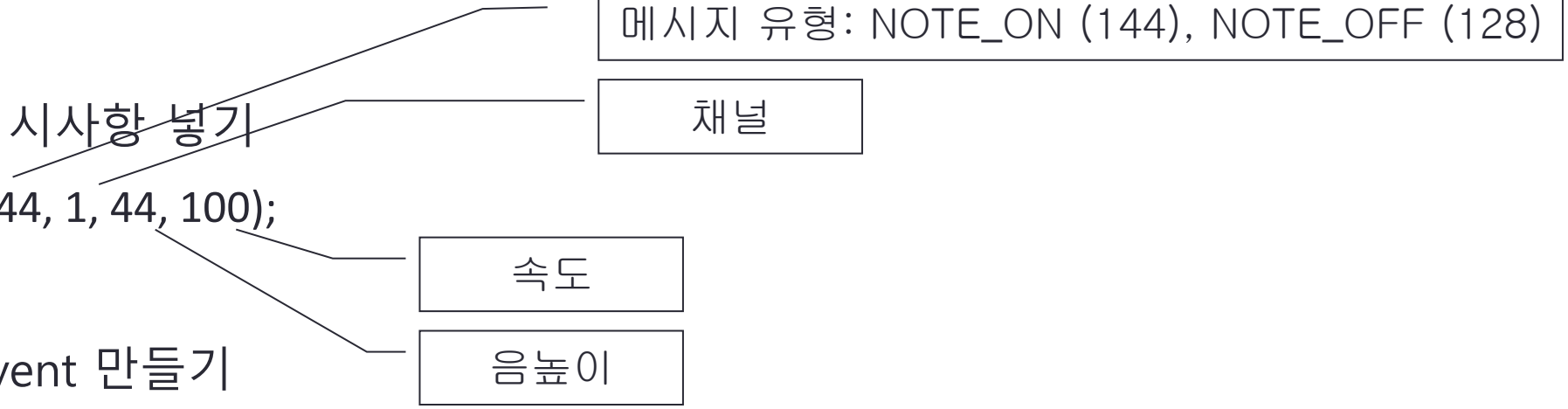
1. Message 만들기

```
ShortMessage a = new ShortMessage();
```

cf. ShortMessage.{NOTE_ON, NOTE_OFF, …}

2. Message에 지시사항 넣기

```
a.setMessage(144, 1, 44, 100);
```



3. 새로운 MidiEvent 만들기

```
MidiEvent noteOn = new MidiEvent(a, 1);
```

cf. 1 tick = 60,000 / (BPM * PPQ) msec

4. MidiEvent를 트랙에 추가

```
track.add(noteOn);
```

pp.491-492

Sequence 클래스

- Sequencer 객체에 의해 연주될 음악 정보를 가지고 있는 자료 구조
- 생성자

```
public Sequence(float divisionType,
                int resolution)
         throws InvalidMidiDataException
```

Parameters:

divisionType - the timing division type (PPQ or one of the SMPTE types)

resolution - the timing resolution

Track 클래스

- 새 Track 객체 생성
  - Sequence.createTrack() 호출로 가능

- 메소드

Modifier and Type	Method and Description
boolean	**add**(MidiEvent event) Adds a new event to the track.
MidiEvent	**get**(int index) Obtains the event at the specified index.
boolean	**remove**(MidiEvent event) Removes the specified event from the track.
int	**size**() Obtains the number of events in this track.
long	**ticks**() Obtains the length of the track, expressed in MIDI ticks.

ShortMessage 클래스

- 생성자

```
public ShortMessage()
```

Constructs a new ShortMessage.

- setMessage()

```
public void setMessage(int command,
                       int channel,
                       int data1,
                       int data2)
                throws InvalidMidiDataException
```

Parameters:

command - the MIDI command represented by this message

channel - the channel associated with the message

data1 - the first data byte

data2 - the second data byte

command용 상수

- 정적 변수

static int	**NOTE_OFF** Command value for Note Off message (0x80, or 128)
static int	**NOTE_ON** Command value for Note On message (0x90, or 144)
static int	**PROGRAM_CHANGE** Command value for Program Change message (0xC0, or 192)

MidiEvent 클래스

- 생성자

MidiEvent(MidiMessage message, long tick)
Constructs a new MidiEvent.

- 메소드

MidiMessage	**getMessage()** Obtains the MIDI message contained in the event.
long	**getTick()** Obtains the time-stamp for the event, in MIDI ticks
void	**setTick(long tick)** Sets the time-stamp for the event, in MIDI ticks

MiniMusicCmdLine.java

```
import javax.sound.midi.*;

public class MiniMusicCmdLine {    // this is the first one

   public static void main(String[] args) {
        MiniMusicCmdLine mini = new MiniMusicCmdLine();
        if (args.length < 2) {
            System.out.println("Don't forget the instrument and note args");
        } else {
            int instrument = Integer.parseInt(args[0]);
            int note = Integer.parseInt(args[1]);
            mini.play(instrument, note);
        }
   } // close 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();
```

p.494

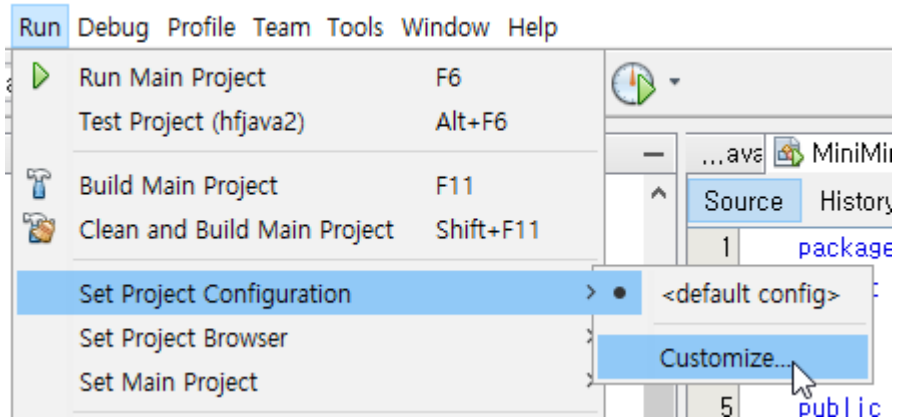
```
            MidiEvent event = null;

            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();}
    } // close play
} // close class
```

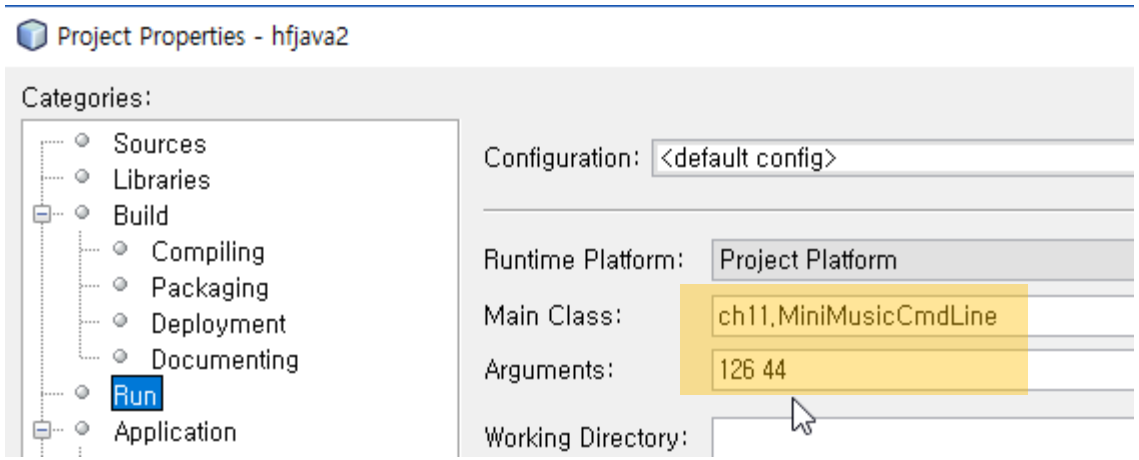
Run it with two int args from 0 to 127. Try these for starters:

```
File Edit Window Help Alternate
%java MiniMusicCmdLine 102 30
%java MiniMusicCmdLine 80 20
%java MiniMusicCmdLine 40 70
```

NetBeans IDE에서의 실행

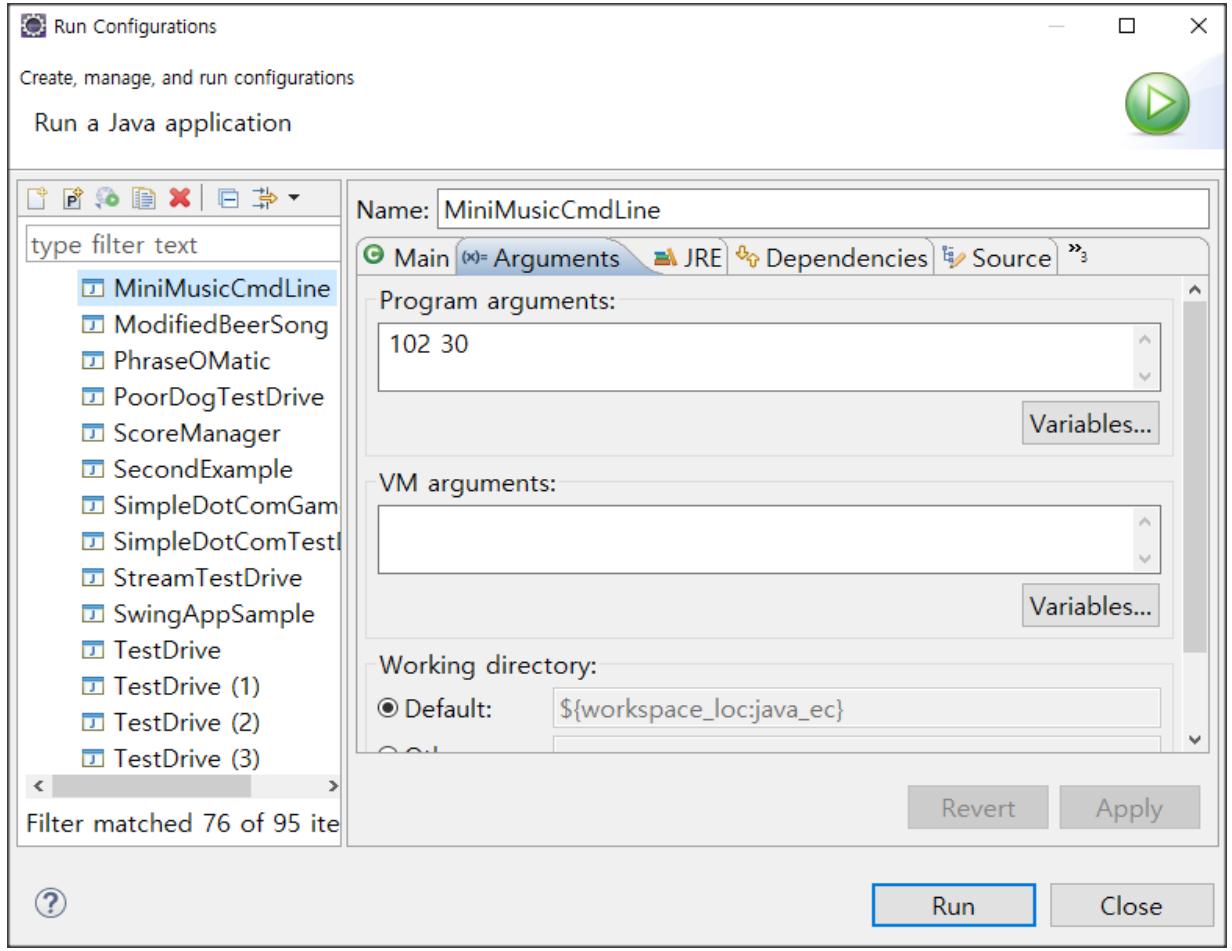
- Run Main Project (F6) 사용하여 실행





Eclipse IDE에서의 실행

- Run > Run Configuration 실행



악기(instrument) 종류 찾아 보기

- 참고 URL: https://stackoverflow.com/questions/30718831/midi-midimessage-program-change-with-instrument-from-different-bank

```
Synthesizer synthesizer = MidiSystem.getSynthesizer();
synthesizer.open();
Instrument instruments = synthesizer.getDefaultSoundbank().getInstruments();
for (Instrument i : instruments)
    System.out.println(i);
```

```
Instrument: Piano 1      bank #0 preset #0
Instrument: Piano 2      bank #0 preset #1
[...]
Instrument: Applause     bank #0 preset #126
Instrument: Gun Shot     bank #0 preset #127
Instrument: SynthBass101 bank #128 preset #38
Instrument: Trombone 2   bank #128 preset #57
[...]
Instrument: Machine Gun  bank #128 preset #127
Instrument: Echo Pan     bank #256 preset #102
Instrument: String Slap  bank #256 preset #120
```

```
Instrument: Lasergun     bank #256 preset #127
[...]
Instrument: Starship     bank #1024 preset #125
Instrument: Carillon     bank #1152 preset #14
[...]
Instrument: Choir Aahs 2 bank #4096 preset #52
```

숙제

- 본문을 꼼꼼하게 읽어봅시다.
- 연필을 깎으며 및 11장 끝에 있는 연습문제를 모두 각자의 힘으로 해결해봅시다.
- API 문서에서 이 장에 나와있는 클래스 및 메소드에 대한 내용을 직접 찾아봅시다.