제 7장. 예외 처리

학습 목표

- 예외의 개념과 예외가 발생하는 상황을 학습합니다.
- 예외 발생 시의 흐름과 처리 방법을 학습합니다.
- 예외를 직접 처리하는 방법과 관련 키워드를 학습합니다.
- 예외를 간접 처리하는 방법과 관련 키워드를 학습합니다.

예외 처리

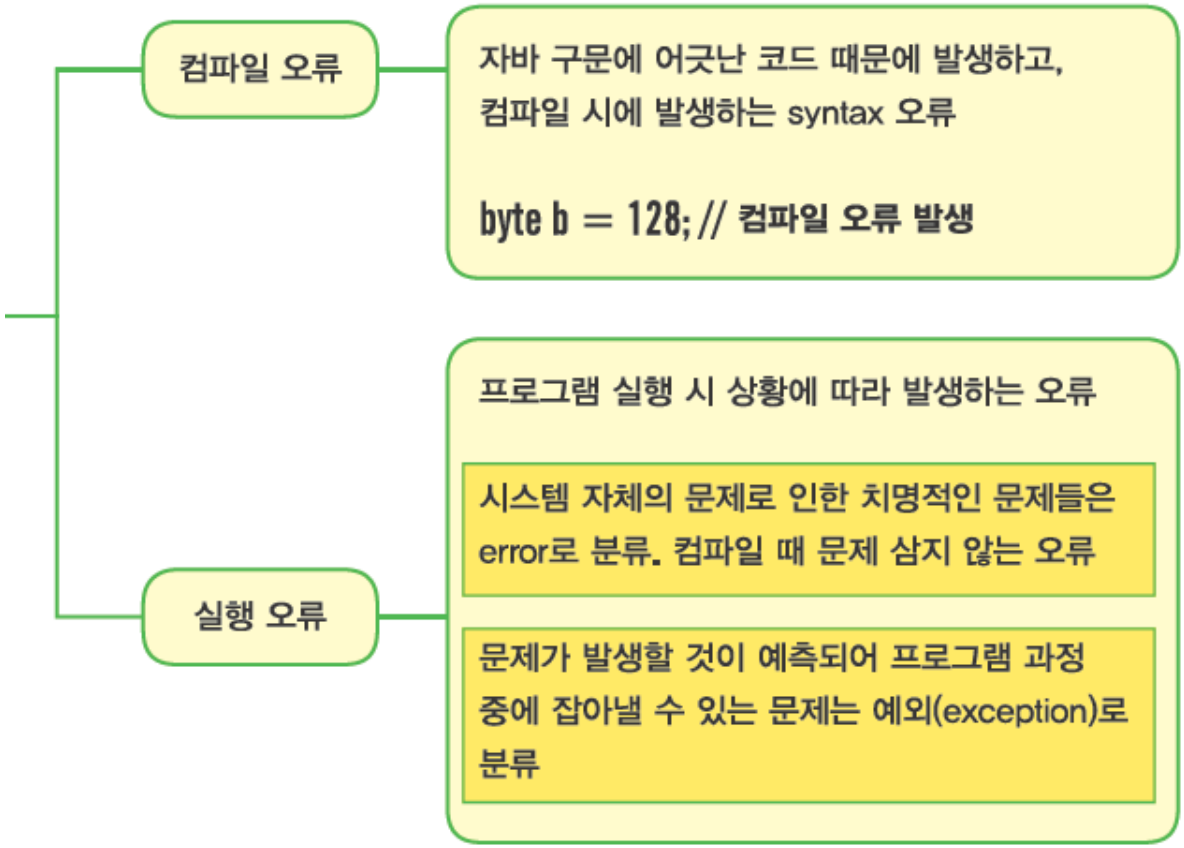
- 예외란
- 예외 클래스
- 예외의 직접 처리
- 예외의 간접 처리

예외란

- 자바 프로그램의 오류들
- 예외와 예외 상황

자바 프로그램의 오류들

* 자바의 오류들



예외와 예외 상황

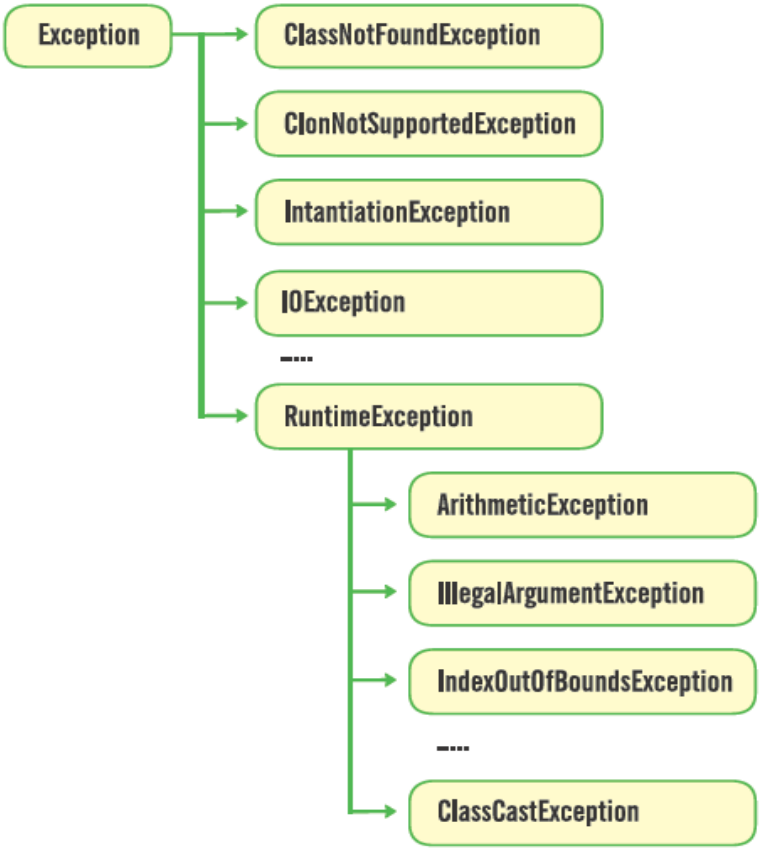
- 프로그램 실행 중 발생할 수 있는 예상치 못한 사건
  - 정수를 0으로 나누는 경우
  - 배열의 인덱스가 배열 길이를 넘어서는 경우
  - 부적절한 형변환이 발생하는 경우
  - 입출력 파일이 존재하지 않는 경우
  - null 값 참조하는 경우 등

예외 클래스

- 예외 상황과 예외 클래스
- 예외 발생시 프로그램의 흐름

예외 상황과 예외 클래스

* 자바의 예외 클래스들

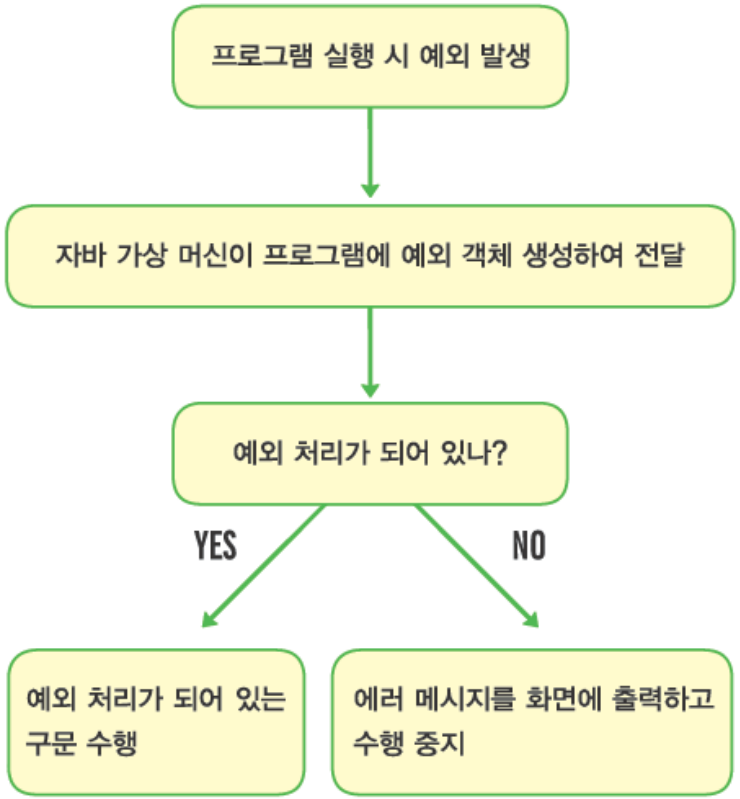


예외 상황과 예외 클래스

- 자바의 예외 클래스들
- java.lang.Exception 클래스 상속
  - 정수를 0으로 나누는 경우 ArithmeticException
  - 배열의 인덱스가 배열 길이를 넘어서는 경우 ArrayIndexOutOfBoundsException
  - 메소드의 매개변수 잘못 지정한 경우 IllegalArgumentException
  - 입출력 파일이 존재하지 않는 경우 IOException

예외 발생시 프로그램의 흐름

* 예외 발생시의 흐름



예외 발생시 프로그램의 흐름

- 프로그램 7-1 실습
  - ExceptionTest.java 작성
  - 예외 발생시 결과 확인
  - 실행 결과

```
C:₩JAVA>java ExceptionTest
x[2] = 0
x[1] = 0
x[0] = 0
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 6
at ExceptionTest.main(ExceptionTest.java:7)
```

예외의 직접 처리

- try-catch 키워드
- finally 키워드

try-catch 키워드

- 예외 발생 감지하여 예외 발생 상황 해결하는 구문

```
문장 1;
try {
    문장 2;
    문장 3;
    문장 N;
}
catch (ExceptionType1  e) {
    복구 루틴 1;
}
catch (ExceptionType2  e) {
    복구 루틴 2;
}
catch (ExceptionTypeM  e) {
    복구 루틴 M;
}
문장 N+1;
```

try-catch 키워드

- 프로그램 7-2 실습
  - TryCatchTest.java 작성
  - 프로그램 7-1 수정하여 예외 발생시 처리 결과 확인
  - 실행 결과

```
7 : 4 이상이면 4로 나눈 나머지로 계산===> x[3] = 0
x[3] = 0
x[1] = 0
x[2] = 0
x[2] = 0
8 : 4 이상이면 4로 나눈 나머지로 계산===> x[0] = 0
5 : 4 이상이면 4로 나눈 나머지로 계산===> x[1] = 0
8 : 4 이상이면 4로 나눈 나머지로 계산===> x[0] = 0
```

try-catch 키워드

- catch 키워드 나열시 순서 중요
- 하위클래스 예외를 상위클래스 예외보다 먼저 catch하도록 지정하지 않으면 오류

```
try {
    System.out.println("x[i] = " + x[i]);
}
catch (Exception  e) {
    System.out.println(e.getMessage());
}
catch (ArrayIndexOutOfBoundsException  e) {
    System.out.print
    (i + " : 4 이상이면 4로 나눈 나머지로 계산===> ");
    i = i % 4;
    System.out.println ("x[" + i + "] = " + x[i]);
}
```

try-catch 키워드

- 프로그램 7-3 실습
  - TryCatchsTest.java 작성
  - 여러가지 예외 처리 결과 확인
  - 발생된 예외 없는 경우 예외 발생하는 경우

```
C:₩JAVA>java TryCatchsTest 10 10
1
완료
```

```
C:₩JAVA>java TryCatchsTest
두 개의 값을 입력하세요
```

```
C:₩JAVA>java TryCatchsTest 10 0
0이 아닌 값으로 입력하세요
```

```
C:₩JAVA>java TryCatchsTest a b
그 밖의 다른 예외가 발생했습니다.
For input string: "a"
완료
```

try-catch 키워드

try-catch 흐름

```
try{
    문장 1;
    문장 2;
}catch(ExceptionA e){
    문장 3;
}catch(ExceptionB e){
    문장 4;
}
문장 5;
```

1. 문장 1에서 예외가 발생하지 않은 경우 수행되는 문장의 흐름

   문장 1 – 문장 2 – 문장 5

2. 문장 1에서 ExceptionA 타입의 예외가 발생하는 경우 수행되는 문장의 흐름

   문장 3 – 문장 5

3. 문장 1에서 ExceptionB 타입의 예외가 발생하는 경우 수행되는 문장의 흐름

   문장 4 – 문장 5

4. 문 장1에서 ExceptionA나 ExceptionB 이외의 다른 타입의 예외가 발생하는 경우(예외 처리(catch)가 안 된 경우) 수행되는 문장의 흐름(예외 처리(catch)가 안 된 경우)

   수행 문장 없음

finally 키워드

- 예외발생 여부나 예외 타입과 무관하게 실행하는 블록

```
try {
    fd = FileOpen("test.txt");
    fd.read();
} catch (IOException e) {
    System.out.println(e.getMessage());
    e.printStackTrace();
} finally {
    CloseFile(fd);
}
```

- finally 블록은 return 키워드 만나도 항상 실행

finally 키워드

- 프로그램 7-4 실습
  - FinallyTest.java 작성
  - 예외 발생시 finally 블록 수행 여부 확인
  - 발생된 예외 없는 경우 예외 발생하는 경우

```
C:₩JAVA>java FinallyTest 10 10
1
항상 출력됩니다.
완료
```

```
C:₩JAVA>java FinallyTest
두개의 값을 입력하세요
C:₩JAVA>java FinallyTest 10 0
0이 아닌 값으로 입력하세요
항상 출력됩니다.
```

```
C:₩JAVA>java FinallyTest a b
그밖의 다른 예외가 발생했습니다.
For input string: "a"
항상 출력됩니다.
완료
```

예외의 간접 처리

- throws 키워드
- 예외 처리가 필요없는 예외들

throws 키워드

- 메소드 내부에서 발생하는 예외를 다른 메소드로 전달하는 역할
- 예외를 전달받은 메소드가 간접 처리
- throws 키워드 사용법

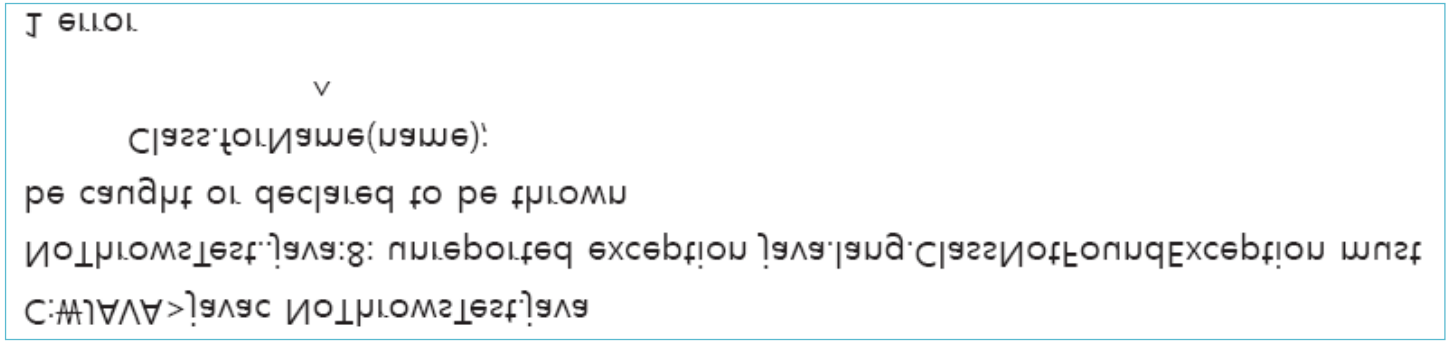
```
[modifier] 리턴타입 메소드이름([매개변수 리스트])
throws 예외클래스1, 예외클래스2 {
    // 메소드 구현부
}
```

- throws 키워드 선언 예

```
public static Class <?> forName(String className)
                    throws ClassNotFoundexception
```

throws 키워드

- 프로그램 7-5 실습
  - NoThrowsTest.java 작성
  - 예외 처리되지 않은 결과 확인
  - 실행 결과



throws 키워드

- 프로그램 7-6 실습
  - ThrowsTest.java 작성
  - 프로그램 7-5를 수정하여 간접 예외 처리
  - 예외 발생하지 않은 경우

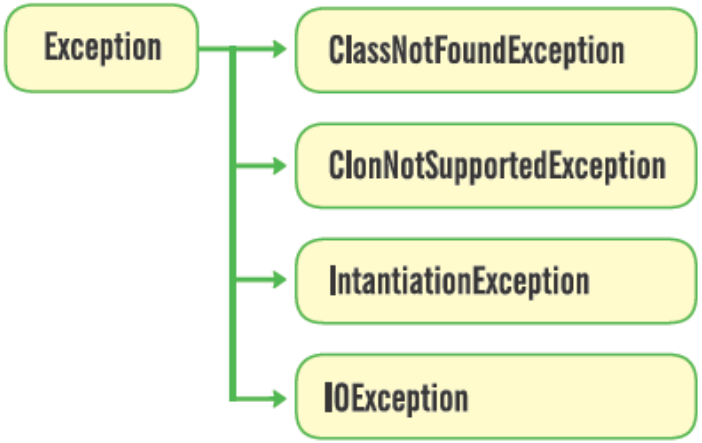
```
C:₩JAVA>java ThrowsTest ThrowsTest
ThrowsTest 클래스는 시스템에 존재합니다.
```

  - 예외 발생 후 처리하는 경우

```
C:₩JAVA>java ThrowsTest A
A 클래스는 시스템에 존재하지 않습니다.
```

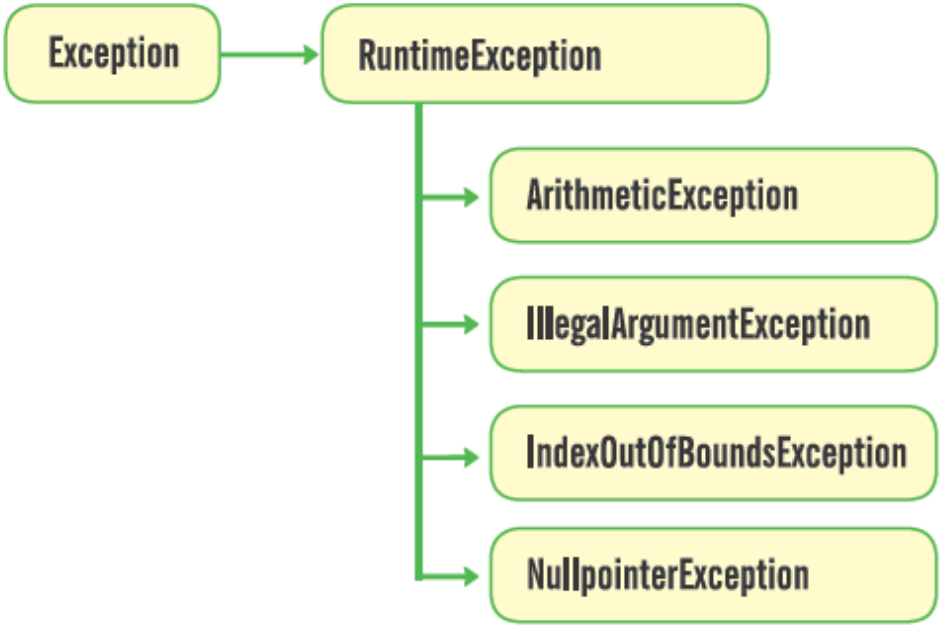
예외 처리가 필요없는 예외들

- 자바 언어는 발생 예외에 대비하여 처리할 것을 경고하는 언어
- java.lang.Exception 클래스에서 상속받은 클래스들은 예외 처리 반드시 필요



예외 처리가 필요없는 예외들

* java.lang.RuntimeException 클래스에서 상속받은 클래스들은 예외 처리 요구하지 않음



정리

- 실행 시의 예외(exception)는 프로그래머의 노력으로 처리할 수 있는 문제들로, 문제가 발생할 것이 예측되어 프로그램 과정 중에 예외를 처리하여 견고한 프로그램을 작성하도록 합니다.
- 자바 언어의 모든 예외는 클래스로 취급하며 이러한 자바의 모든 예외 클래스들은 java.lang 패키지의 Exception 클래스에서 상속받은 클래스들입니다.
- 자바 프로그램 실행 시에 예외가 발생하면 JVM이 발생한 예외 객체를 생성하여 넘겨주고 적절한 에러 메시지를 출력한 다음에 프로그램은 중단됩니다.

정리

- 자바 프로그램의 예외 처리는 예외가 발생한 메소드 내부에서 처리하는 try-catch-finally 블록을 이용한 처리 방법과 예외가 발생한 메소드에서 예외를 다른 곳으로 전달하는 throws 키워드를 이용한 처리 방법이 있습니다.
- try-catch-finally 블록의 구조

```
try {
예외가 발생할 가능성이 있는 문장들;
}catch (ExceptionType e) {
ExceptionType 예외 발생 시 복구 문장들;
}finally {
예외 발생 여부나 예외 타입과 관계없이 항상 실행할 문장들;
}
```

정리

- throws 키워드를 이용한 선언은 다른 메소드로 예외를 전달하여 전달받은 다른 메소드에서 처리하도록 하는 방법으로 메소드 선언부에 선언합니다.
- RuntimeException 클래스에서 상속받은 예외의 경우 주로 프로그래머의 부주의로 인해 발생하는 것으로, 자바 컴파일러는 이런 예외 발생 상황을 프로그램에서 처리하기를 요구하지 않습니다.