



2024학년도 1학기

신입생 Java 교육

교육부장 20 이어진

교육 커리큘럼

1주차 변수, 연산자 + 조건문(if)

2주차 조건문(switch) + 반복문

3주차 배열

<중간고사>

4주차 객체, 메소드 오버로딩과 생성자

5주차 상속과 오버라이딩

6주차 다형성과 인터페이스

7주차 예외처리

<기말고사>



Part 1

예외처리



Part 1

프로그램 오류

컴파일 에러 - 컴파일 할 때 발생하는 에러
런타임 에러 - 실행할 때 발생하는 에러

- 에러(error)는 어쩔 수 없지만, 예외(exception)는 처리해야 한다.

에러(error) - 프로그램 코드에 의해서 수습될 수 없는 심각한 오류

예외(exception) - 프로그램 코드에 의해서 수습될 수 있는 다소 미약한 오류

- 예외처리의 정의와 목적

예외처리(exception handling)의

정의 - 프로그램 실행 시 발생할 수 있는 예외의 발생에 대비한 코드를 작성하는 것

목적 - 프로그램의 비정상 종료를 막고, 정상적인 실행상태를 유지하는 것

[참고] 에러와 예외는 모두 실행 시(runtime) 발생하는 오류이다.

Part 1

예외처리구문(try-catch)

예외를 처리하려면 try-catch문을 사용해야 한다.

```
try {  
    // 예외가 발생할 가능성이 있는 문장들을 넣는다.  
} catch (Exception1 e1) {  
    // Exception1이 발생했을 경우, 이를 처리하기 위한 문장을 적는다.  
} catch (Exception2 e2) {  
    // Exception2가 발생했을 경우, 이를 처리하기 위한 문장을 적는다.  
    ...  
} catch (ExceptionN eN) {  
    // ExceptionN이 발생했을 경우, 이를 처리하기 위한 문장을 적는다.  
}
```

[참고] if문과 달리 try블럭이나 catch블럭 내에 포함된 문장이 하나라고 해서 괄호{}를 생략할 수는 없다.

▶ try블럭 내에서 예외가 발생한 경우,

1. 발생한 예외와 일치하는 catch블럭이 있는지 확인한다.
2. 일치하는 catch블럭을 찾게 되면, 그 catch블럭 내의 문장들을 수행하고 전체 try-catch문을 빠져나가서 그 다음 문장을 계속해서 수행한다. 만일 일치하는 catch블럭을 찾지 못하면, 예외는 처리되지 못한다.

▶ try블럭 내에서 예외가 발생하지 않은 경우,

1. catch블럭을 거치지 않고 전체 try-catch문을 빠져나가서 수행을 계속한다.

Part 1 try-catch 문 예시

```
class ExceptionEx4 {  
    public static void main(String args[]) {  
        System.out.println(1);  
        System.out.println(2);  
  
        try {  
            System.out.println(3);  
            System.out.println(4);  
        } catch (Exception e) {  
            System.out.println(5);  
        } // try-catch의 끝  
        System.out.println(6);  
    } // main메서드의 끝  
}
```

[실행결과]

1
2
3
4
6

```
class ExceptionEx5 {  
    public static void main(String args[]) {  
        System.out.println(1);  
        System.out.println(2);  
        try {  
            System.out.println(3);  
            System.out.println(0/0);  
            System.out.println(4);  
        } catch (ArithmeticException ae) {  
            System.out.println(5);  
        } // try-catch의 끝  
        System.out.println(6);  
    } // main메서드의 끝  
}
```

[실행결과]

1
2
3
5
6

Part 1

예외 발생시키기

1. 먼저, 연산자 `new`를 이용해서 발생시키려는 예외 클래스의 객체를 만든 다음

```
Exception e = new Exception("고의로 발생시켰음");
```

2. 키워드 `throw`를 이용해서 예외를 발생시킨다.

```
throw e;
```

[예제8-6]/ch8/ExceptionEx6.java

```
class ExceptionEx6
{
    public static void main(String args[])
    {
        try {
            Exception e = new Exception("고의로 발생시켰음.");
            throw e; // 예외를 발생시킴
            // throw new Exception("고의로 발생시켰음.");
        } catch (Exception e) {
            System.out.println("에러 메시지 : " + e.getMessage());
            e.printStackTrace();
        }
        System.out.println("프로그램이 정상 종료되었음.");
    }
}
```

위의 두 줄을 한 줄로
줄여 쓸 수 있다.

[실행결과]

```
에러 메시지 : 고의로 발생시켰음.
java.lang.Exception: 고의로 발생시켰음.
    at ExceptionEx6.main(ExceptionEx6.java:6)
프로그램이 정상 종료되었음.
```

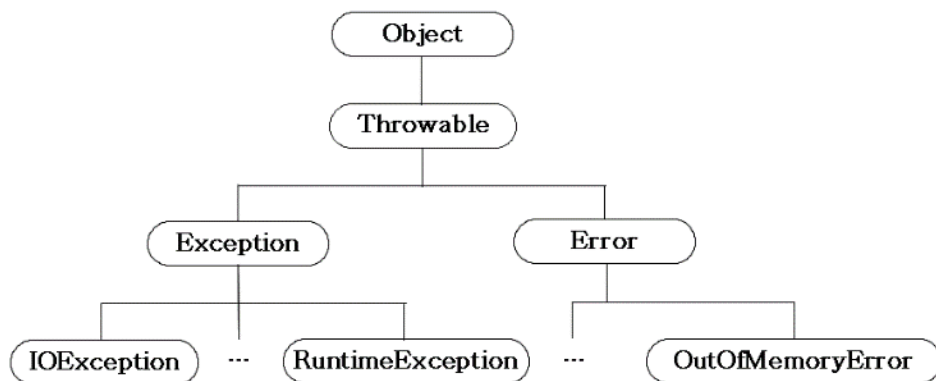
Part 1

예외 클래스의 계층구조

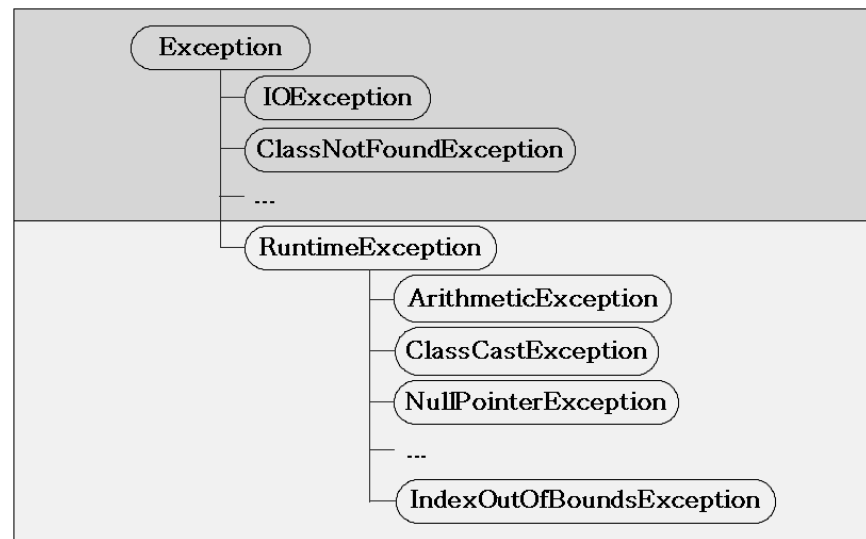
예외 클래스는 크게 두 그룹으로 나뉜다.

RuntimeException 클래스들 - 프로그래머의 실수로 발생하는 예외 ← 예외처리 필수

Exception 클래스들 - 사용자의 실수와 같은 외적인 요인에 의해 발생하는 예외 ← 예외처리 선택



[그림8-1] 예외 클래스 계층도



[그림8-2] Exception 클래스와 RuntimeException 클래스 중심의 상속계층도

Part 1

일반 예외 (Exception 클래스)

NoSuchMethodException	메소드가 존재하지 않을 때
ClassNotFoundException	클래스가 존재하지 않을 때
CloneNotSupportedException	객체의 복제가 지원되지 않는 상황에서 복제를 시도하고자하는 경우
IllegalAccessException	클래스에 대한 부정적인 접근
InstantiationException	추상클래스나 인터페이스로부터 객체를 생성하고자 하는 경우
InterruptedException	스레드가 인터럽트 되었을 때
RuntimeException	실행시간에 예외가 발생한 경우
IOException	입출력과 관련된 예외 처리

Part 1

실 행 예 외 (Runtime Exception 클 래 스)

ArithmeticException	0으로 나누는 등의 산술적인 예외
NegativeArraySizeException	배열의 크기를 지정할 때 음수의 사용
NullPointerException	Null 객체의 메소드나 멤버 변수에 접근하고자 하는 경우
IndexOutOfBoundsException	배열이나 스트링 범위를 벗어날 때
SecurityException	보안을 이유로 메소드를 수행할 수 없을때

InputMismatchException : 정수로 입력해야 하는데 문자를 입력한 경우

Part 1

예외의 발생과 catch블럭

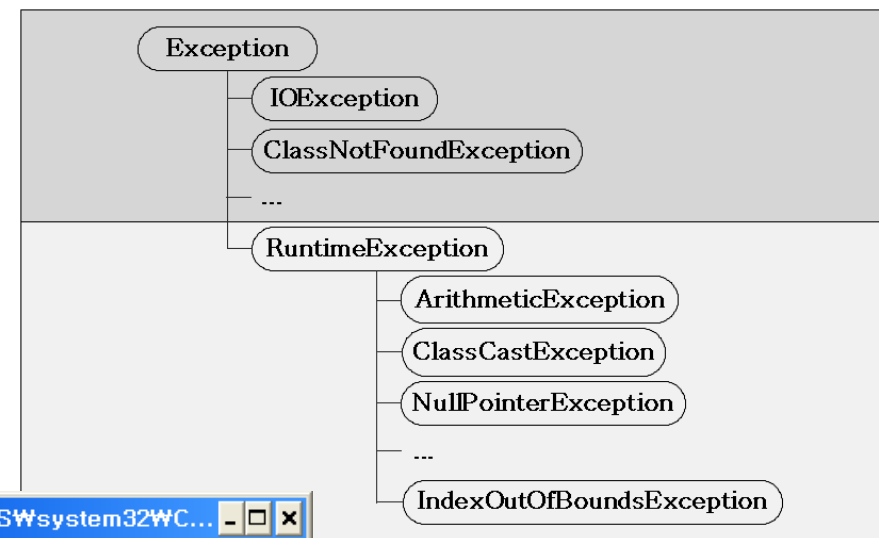
- try블럭에서 예외가 발생하면, 발생한 예외를 처리할 catch블럭을 찾는다.
- 첫번째 catch블럭부터 순서대로 찾아 내려가며, 일치하는 catch블럭이 없으면 예외는 처리되지 않는다.
- 예외의 최고 조상인 Exception을 처리하는 catch블럭은 모든 종류의 예외를 처리할 수 있다.(반드시 마지막 catch블럭이어야 한다.)

[예제8-11]/ch8/ExceptionEx11.java

```
class ExceptionEx11 {  
    public static void main(String args[]) {  
        System.out.println(1);  
        System.out.println(2);  
        try {  
            System.out.println(3);  
            System.out.println(0/0);  
            System.out.println(4);  
        } catch (ArithmeticException ae) {  
            if (ae instanceof ArithmeticException)  
                System.out.println("true");  
            System.out.println("ArithmeticException");  
        } catch (Exception e) {  
            System.out.println("Exception");  
        }  
        // try-catch의 끝  
        System.out.println(6);  
    }  
    // main메서드의 끝  
}
```

0으로 나눠서
ArithmeticException을
발생시킨다.

ArithmeticException을
제외한 모든 예외가 처리된
다.



```
C:\W\jdk1.5\work>java ExceptionEx11  
1  
2  
3  
true  
ArithmeticException  
6
```

RuntimeException클래스 중심의 상속계통도

Part 1

예외의 발생과 catch블럭

- 발생한 예외 객체를 catch블럭의 참조변수로 접근할 수 있다.

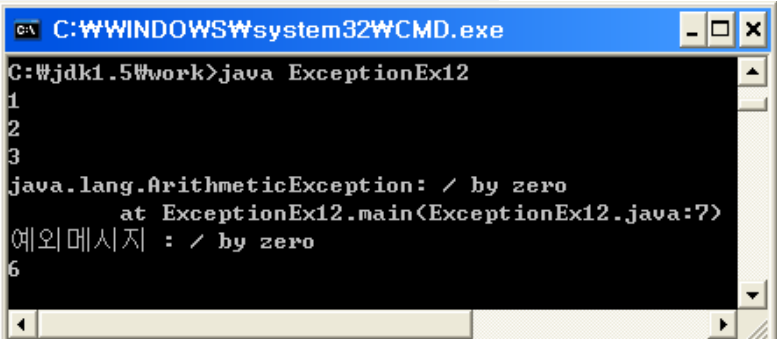
printStackTrace() - 예외발생 당시의 호출스택(Call Stack)에 있었던 메서드의 정보와 예외 메시지를 화면에 출력한다.

getMessage() - 발생한 예외클래스의 인스턴스에 저장된 메시지를 얻을 수 있다.

[예제8-12]/ch8/ExceptionEx12.java

```
class ExceptionEx12 {  
    public static void main(String args[]) {  
        System.out.println(1);  
        System.out.println(2);  
        try {  
            System.out.println(3);  
            System.out.println(0/0); // 예외발생!!!  
            System.out.println(4); // 실행되지 않는다.  
        } catch (ArithmeticException ae) {  
            ae.printStackTrace();  
            System.out.println("예외메시지 : " + ae.getMessage());  
        } // try-catch의 끝  
        System.out.println(6);  
    } // main메서드의 끝  
}
```

참조변수 ae를 통해, 생성된
ArithmeticException인
스턴스에 접근할 수 있다.



```
C:\WINDOWS\system32\CMD.exe  
C:\w\jdk1.5\work>java ExceptionEx12  
1  
2  
3  
java.lang.ArithmeticException: / by zero  
    at ExceptionEx12.main(ExceptionEx12.java:7)  
예외메시지 : / by zero  
6
```

Part 1

finally 블록

- 예외의 발생여부와 관계없이 실행되어야 하는 코드를 넣는다.
- 선택적으로 사용할 수 있으며, try-catch-finally의 순서로 구성된다.
- 예외 발생시, try → catch → finally의 순서로 실행되고
예외 미발생시, try → finally의 순서로 실행된다.
- try 또는 catch블록에서 return문을 만나도 finally블록은 수행된다.

```
try {  
    // 예외가 발생할 가능성이 있는 문장들을 넣는다.  
} catch (Exception1 e1) {  
    // 예외처리를 위한 문장을 적는다.  
} finally {  
    // 예외의 발생여부에 관계없이 항상 수행되어야하는 문장들을 넣는다.  
    // finally블록은 try-catch문의 맨 마지막에 위치해야한다.  
}
```

Part 1 finally 블록 예시

```
public class Main {  
    Run | Debug  
    public static void main(String[] args) {  
        try {  
            System.out.println("try 문 (예외 발생 x)");  
        } catch (Exception e) {  
            System.out.println("catch 문 (예외 발생 x)");  
            return;  
        } finally {  
            System.out.println("finally 문 (예외 발생 x)");  
        }  
        System.out.println("=====");  
        try {  
            System.out.println("try 문");  
            throw new Exception("예외 발생");  
        } catch (Exception e) {  
            System.out.println("catch 문");  
            return;  
        } finally {  
            System.out.println("finally 문");  
        }  
    }  
}
```

```
try 문 (예외 발생 x)  
finally 문 (예외 발생 x)  
=====  
try 문  
catch 문  
finally 문
```

실습문제

[문제]

두 개의 정수를 입력 받아 나눗셈을 하는 프로그램을 만들어보자.

[조건]

정수를 입력하지 않았을 때와 0으로 나눌 때에 대한 예외처리 필요.

[출력 예시]

```
어떤 수를 나누시겠습니까? : d
정수를 입력해주세요. 다시 입력해주세요.
어떤 수를 나누시겠습니까? : 10
어떤 수로 나누시겠습니까? : d
정수를 입력해주세요. 다시 입력해주세요.
어떤 수로 나누시겠습니까? : 0
0으로 나눌 수 없습니다. 다시 입력해주세요.
어떤 수로 나누시겠습니까? : 5
10 / 5 = 2
```

해설

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a, b;
        while(true) {
            try {
                System.out.print("어떤 수를 나누시겠습니까? : ");
                a = sc.nextInt();
                break;
            } catch ( ) {
                System.out.println("정수를 입력해주세요. 다시 입력해주세요.");
                sc = new Scanner(System.in);
            }
        }
        while(true) {
            try {
                System.out.print("어떤 수로 나누시겠습니까? : ");
                b = sc.nextInt();
                if (b == 0) {
                    throw new ;
                }
                break;
            } catch ( ) {
                System.out.println("0으로 나눌 수 없습니다. 다시 입력해주세요.");
                sc = new Scanner(System.in);
            } catch ( ) {
                System.out.println("정수를 입력해주세요. 다시 입력해주세요.");
                sc = new Scanner(System.in);
            }
        }
        System.out.println(a + " / " + b + " = " + a / b);
    }
}
```

```
어떤 수를 나누시겠습니까? : d
정수를 입력해주세요. 다시 입력해주세요.
어떤 수를 나누시겠습니까? : 10
어떤 수로 나누시겠습니까? : d
정수를 입력해주세요. 다시 입력해주세요.
어떤 수로 나누시겠습니까? : 0
0으로 나눌 수 없습니다. 다시 입력해주세요.
어떤 수로 나누시겠습니까? : 5
10 / 5 = 2
```


해설

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a, b;
        while(true) {
            try {
                System.out.print("어떤 수를 나누시겠습니까? : ");
                a = sc.nextInt();
                break;
            } catch (InputMismatchException e) { <- Exception e 로 바뀌도 됨
                System.out.println("정수를 입력해주세요. 다시 입력해주세요.");
                sc = new Scanner(System.in);
            }
        }
        while(true) {
            try {
                System.out.print("어떤 수로 나누시겠습니까? : ");
                b = sc.nextInt();
                if (b == 0) {
                    throw new ArithmeticException();
                }
                break;
            } catch (ArithmeticException e) {
                System.out.println("0으로 나눌 수 없습니다. 다시 입력해주세요.");
                sc = new Scanner(System.in);
            } catch (InputMismatchException e) { <- Exception e 로 바뀌도 됨
                System.out.println("정수를 입력해주세요. 다시 입력해주세요.");
                sc = new Scanner(System.in);
            }
        }
        System.out.println(a + " / " + b + " = " + a / b);
    }
}
```

```
어떤 수를 나누시겠습니까? : d
정수를 입력해주세요. 다시 입력해주세요.
어떤 수를 나누시겠습니까? : 10
어떤 수로 나누시겠습니까? : d
정수를 입력해주세요. 다시 입력해주세요.
어떤 수로 나누시겠습니까? : 0
0으로 나눌 수 없습니다. 다시 입력해주세요.
어떤 수로 나누시겠습니까? : 5
10 / 5 = 2
```

Q&A



감사합니다