예외처리

안 화 수

예외처리

- 예외

  프로그램이 실행되는 동안에 발생하는 예상하지 못한 않은 에러를 의미한다.

- 예외처리를 하는 이유

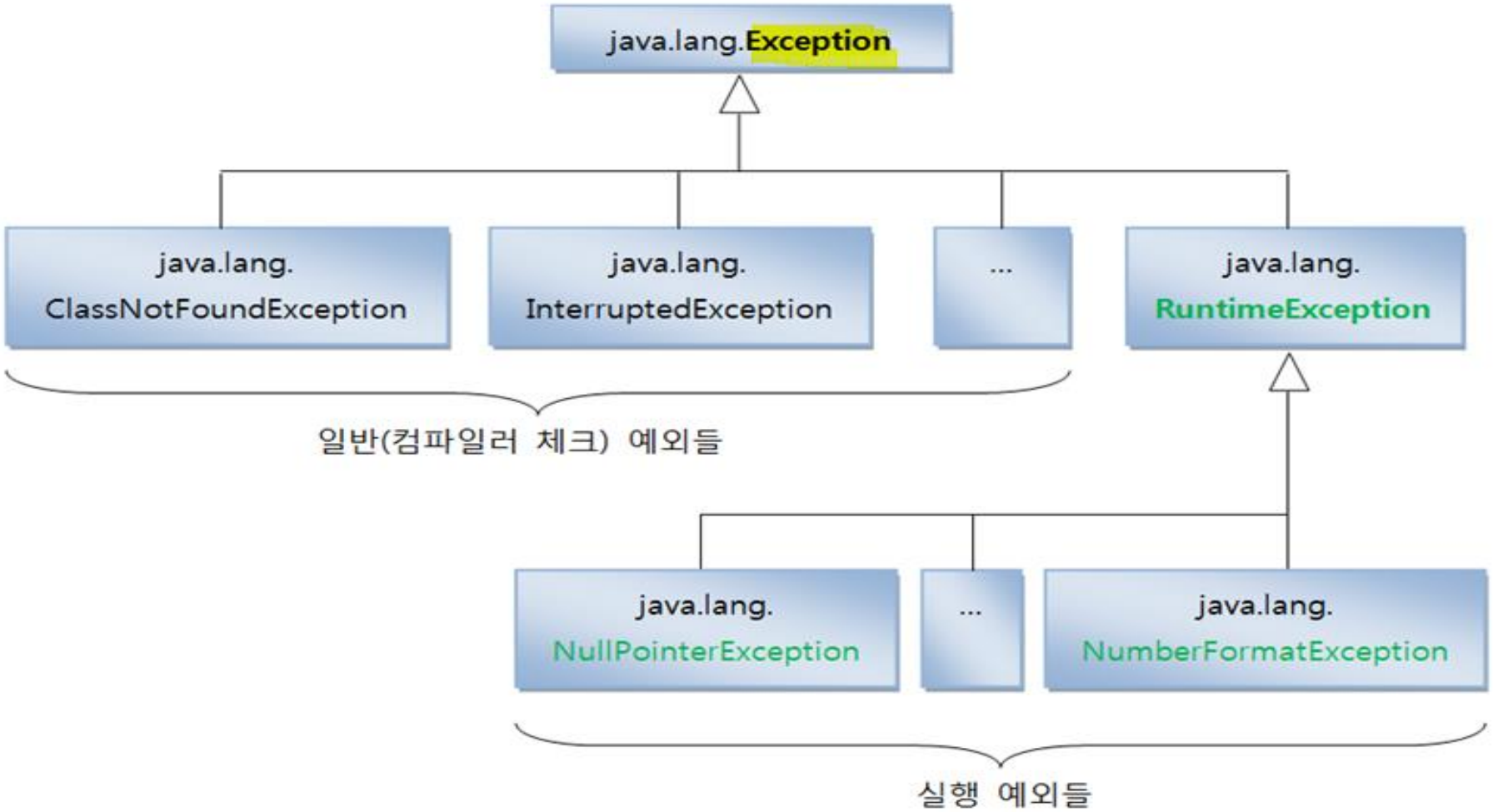
  프로그램을 안전하게 종료 시키기 위해서 예외처리를 한다.

- 예외처리를 해야되는 경우

  1. 입출력을 처리할 때
  2. 데이터베이스 연동을 처리할 때

예외처리

- 예외 클래스


java.lang.Exception
java.lang.
ClassNotFoundException
java.lang.
InterruptedException
...
java.lang.
RuntimeException
일반(컴파일러 체크) 예외들
java.lang.
NullPointerException
...
java.lang.
NumberFormatException
실행 예외들

예외처리

- 자주 사용하는 예외 클래스

- ArithmeticException : 0으로 나눌 때 발생
- ArrayIndexOutOfBoundsException : 배열의 범위를 벗어 났을 때 발생
- NumberFormatException : 숫자 외의 값을 사용할 때 발생
- NullPointerException : 객체를 생성하지 않고 필드나 메소드를 호출할 때 발생
- IOException : 입출력을 할 때 발생
- FileNotFoundException : 파일이 없을 때 발생
- ClassCastException : 자료형 변환이 되지 않을 경우에 발생
- SQLException : 데이터베이스와 연동할 때 발생

예외처리

- 예외 처리 형식1.

```
try{
        예외가 발생할 가능성이 있는 문장;
 }catch(예외클래스명   매개변수){
        예외 메시지;
 }
```

예외처리

- 예외 처리 예제.

  예외처리를 하지 않은 경우에는 프로그램이 비정상적으로 종료된다.

```
public class DivideZeroException {

    public static void main(String[] args) {

        // int type의 변수 선언
        int b = 20;
        int a = 0;

        // 두 수의 나눗셈 결과를 구한다
        int c = b / a;              // 예외발생
        int total = a + b;
        int sub = b - a;

        System.out.println(c);
    }
}
```

예외처리

- 예외 처리 예제.

  예외가 발생하면, 예외가 발생한 라인 아랫쪽은 실행되지 않는다.

```
public class DivideZeroExceptionHandling {
    public static void main(String[] args) {
        int b = 20;
        int a = 0;
        int c = 0;

        try {
            c = b / a;              // 예외발생

            //예외가 발생하면, 예외가 발생한 라인 아랫쪽은 실행되지 않는다.
            int total = a + b;
            int sub = b - a;
        } catch (ArithmeticException ae) {
            a = 2;
            c = b / a;
        }
        System.out.println(c);
    }
}
```

예외처리

- 예외 처리 예제.

예외가 발생하면, 모든 예외를 Exception 클래스가 받아서 처리할 수 있다.

```
public class DivideZeroExceptionHandling1 {
    public static void main(String[] args) {

        int b = 20;
        int a = 0;
        int c = 0;

        try {
            c = b / a;                    // 예외발생
            int total = a + b;
            int sub = b - a;
        } catch (Exception e) {
            a = 2;
            c = b / a;
            System.out.println("0으로 나눌수 없습니다.");
        }
        System.out.println(c);
    }
}
```

예외처리

- 예외 처리 예제. 예외가 메시지 출력하기

```
public class DivideZeroExceptionHandling2 {
    public static void main(String[] args) {

        int b = 20;
        int a = 0;
        int c = 0;

        try {
            c = b / a;                          // 예외발생

            // 예외가 발생하면, 예외가 발생한 아랫쪽의 실행되지 않는다.
            int total = a + b;
            int sub = b - a;
            System.out.println(total);
            System.out.println(sub);
        } catch (Exception e) {
            // 예외 정보를 자세히 출력
            e.printStackTrace();

            // 예외 정보를 문자열로 반환함.
            System.out.println(e.getMessage());

            // 예외의 간단한 설명문을 반환함.
            System.out.println(e.toString());

            System.out.println("0으로 나눌수 없습니다.");
        }
    }
}
```

예외처리

- 예외 처리 형식2.

```
try{
        예외가 발생할 가능성이 있는 문장;
}catch(예외클래스명 매개변수){
        예외 메시지;
}catch(예외클래스명 매개변수){
        예외 메시지;
}
```

예외처리

- 예외 처리 예제. 멀티 예외처리

```
public class MultiExceptionHandling {
    public static void main(String[] args) {
        int value = 20;
        int div = 0;
        int[] intArray = { 1, 2, 3 };
        try {
            // int arrayValue = intArray[4];     // 예외발생
            // System.out.println(arrayValue);

            int result = value / div;            // 예외발생
            System.out.println(result);

            // 배열의 특정 값을 저장
            int arrayValue = intArray[4];        // 예외발생
            System.out.println(arrayValue);
        } catch (ArithmeticException ae) {
            System.out.println(ae.toString());
            System.out.println("0으로 나눌수 없습니다.");
        } catch (ArrayIndexOutOfBoundsException ai) {
            ai.printStackTrace();
            System.out.println("배열의 범위를 벗어 났습니다.");
        }
    }
}
```

예외처리

- 예외 처리 예제6. Exception은 가장 마지막에 사용해야 한다.

```
public class ExceptionEx3 {
	public static void main(String[] args) {
		int var = 50;

		try {
			int data = Integer.parseInt(args[0]);
			System.out.println(var / data);

		// Exception 예외 클래스가 하위 예외 클래스들을
		// 모두 가지고 있기 때문에 먼저 마지막에 정의해야 한다.
//		} catch (Exception e) {
//			System.out.println("Exception !!");
		} catch (NumberFormatException ne) {
			System.out.println("숫자가 아닙니다.");
		} catch (ArithmeticException ae) {
			System.out.println("0으로 나눌순 없죠?");
		} catch (Exception e) {
			System.out.println("Exception !!");
		}
		System.out.println("프로그램 종료!");
	}
}
```

예외처리

❖ 예외 처리 형식3.

```
try{
        예외가 발생할 가능성이 있는 문장;
}catch(예외클래스명 매개변수){
        예외 메시지;
}finally{
         //주로 파일을 닫을 때나 데이터베이스 연결 끊을 때 사용됨
         예외가 발생하든, 발생하지 않든 무조건 실행됨;
}
```

예외처리

- 예외 처리 예제.

```
public class MultiExceptionHandling1 {
    public static void main(String[] args) {
        int value = 20;
        int div = 10;

        int[] intArray = { 1, 2, 3 };
        try {
            int result = value / div;              // 예외발생
            System.out.println(result);

            int arrayValue = intArray[2];          // 예외발생
            System.out.println(arrayValue);
        } catch (ArithmeticException ae) {
            System.out.println(ae.toString());
        } catch (ArrayIndexOutOfBoundsException ai) {
            ai.printStackTrace();
        } finally {
            System.out.println("예외가 발생했음!");
        }
    }
}
```

예외처리

❖ 예외 처리 형식4. 메소드를 호출한 곳에 예외를 떠넘기기

```
public class Test{
    public void check()  throws  Exception{
    }
    public static void main(String[] args){
        try{
            Test t = new Test();
            t.check();
        }catch(Exception e){
        }
    }
}
```

예외처리

- 예외 처리 예제. try ~ catch로 처리

```
public class ThrowsException {
    public void occurException() {
        try {
            int result = 3 / 0;        // 예외 발생
            System.out.println(result);
        } catch (ArithmeticException ae) {
            System.out.println("0으로 나눌 수 없습니다.");
        }
    }
    public static void main(String[] args) {
        ThrowsException te = new ThrowsException();

        te.occurException();
    }
}
```

예외처리

- 예외 처리 예제. throws 로 처리

```
public class ThrowsExceptionHandling1 {
    // occurException()를 호출한 곳으로 예외처리를 양도 하겠다는 의미
    public void occurException() throws ArithmeticException {

        int result = 3 / 0;          // 예외발생
        System.out.println(result);
    }

    public static void main(String[] args) {
        ThrowsExceptionHandling1 te = new ThrowsExceptionHandling1();

        try {
            te.occurException();
        } catch (ArithmeticException ae) {
            System.out.println("Exception이 발생 : " + ae.toString());
            System.out.println("0으로 나눌 수 없습니다.");
        }
    }
}
```

예외처리

- 예외 처리 예제.

```
public class ThrowsEx1 {
    public void setData(String n) throws NumberFormatException {
        if (n.length() >= 1) {                          // String n="5"
            String str = n.substring(0, 1);             // String str="5"
            printData(str);
        }
    }
    private void printData(String n) throws NumberFormatException {
        int dan = Integer.parseInt(n);          // String n="5"
        System.out.println(dan + "단");          // 예외발생
        System.out.println("----------");
        for (int i = 1; i < 10; i++)
            System.out.println(dan + "*" + i + "=" + (dan * i));
    }
    public static void main(String[] args) {
        ThrowsEx1 t1 = new ThrowsEx1();
                    // args[0] = "a"    예외발생
                    // args[0] = "5"    예외가 발생 하지 않음
        try {
            t1.setData(args[0]);
        } catch (Exception e) {
            System.out.println("첫문자가 숫자가 아닙니다.");
        }
    }
}
```

예외처리

❖ 예외 처리 형식5. 강제로 예외 발생 시키기

```
public class Test{
        public void check() throws ArrayIndexOutOfBoundsException {
                for( int i=0 ; i<10 ; i++ ) {
                        if( i == 3 ){  // 프로그래머가 예외를 발생시킴
                                throw  new  ArrayIndexOutOfBoundsException();
                        }
                }
        }
        public static void main( String[] args ) {
                Test t = new Test();
            try {
                t.check();
            } catch ( ArrayIndexOutOfBoundsException ab ) {
                ab.printStackTrace();
            }
        }
}
```

예외처리

- 예외 처리 예제. 강제로 예외 발생 시키기

```
public class ThrowException {
    public void exceptionMethod() throws ArrayIndexOutOfBoundsException {
        int[] intA = { 1, 2, 3, 4 };

        for (int i = 0; i < 10; i++) {

            if (i == 2)      // 강제로 예외를 발생 시킴
                throw new ArrayIndexOutOfBoundsException();
            System.out.println(intA[i]);
        }
    }
    public static void main(String[] args) {
        ThrowException te = new ThrowException();

        try {
            te.exceptionMethod();
        } catch (ArrayIndexOutOfBoundsException ab) {
            System.out.println("배열의 index를 초과했습니다.");
            ab.printStackTrace();
        }
    }
}
```