클릭하세요 자바

예외처리

– 예외처리

최문환

예외처리

1. 예외란?
2. 프로그램이 갑자기 죽지 않게 하는 법
3. 다양한 종류의 예외
4. 예외의 인위적인 발생
5. 호출 메서드에 예외 전달
6. 사용자 정의 예외

1. 예외란?

예외란 일반적이지 않는 상황 즉, 프로그램이 실행되는 동안에 발생하는 예기치 않은 에러를 의미합니다.

자바 프로그래머들이 예상하지 못한 상황에 대해서 어떻게 대처할지에 대해서 프로그램을 해두어야 합니다.

<예제> 0으로 나누어 고의적으로 예외를 발생시킨 프로그램

```
001:public class ExcepTest01 {
002:  public static void main(String[] args) {
 003:    int a=10, b01=0, b02=2, c=10;
004:    c = a / b02;  //(1)
005:    System.out.println(" c -> " + c);
006:    c = a / b01; //(2)
007:    System.out.println(" c -> " + c);
008:    c = a / b02; //(3)
009:    System.out.println(" c -> " + c);
010:    System.out.println(" 정상 종료 " + c);
011:}
```


2. 프로그램이 갑자기 죽지 않게 하는 법

★ try-catch와 Exception 클래스

```
try{
   예외가 발생할 만한 코드
 }catch(해당_Exception e) {
   예외 처리를 위한 루틴
 }
```


<예제> try catch 구문으로 예외 처리

```
001:public class ExcepTest03 {
002:   public static void main(String[] args) {
003:     int a=10, b01=0, b02=2, c=10;
004:     System.out.println("try 구문 수행하기 전 c -->" + c);
005:     try {
006:       System.out.println("try 구문으로 들어옴");
007:       System.out.println("나누기 연산하기 전 try 구문");
008:
009:       c = a / b02;  //(1)
010:       System.out.println(" (1) c -> " + c);
011:
012:       c = a / b01; //(2)
013:       System.out.println(" (2) c -> " + c);
014:
015:       c = a / b02; //(3)
016:       System.out.println(" (3) c -> " + c);
017:     }
```

<예제> try catch 구문으로 예외 처리

```
018:    catch(Exception e) {
019:      System.out.println("예외가 발생하여 Exception 객체가 잡음->" + e);
020:    }//예외처리 끝
021:    System.out.println("try 구문을 수행한 후 c-->" + c);
022:  }
023:}
```


<예제> 하나의 try문과 여러 개의 catch 블록

```
001:public class ExcepTest04 {
002:   public static void main(String[] args) {
003:     int a=10, b01=0, b02=2, c=10;
004:     System.out.println("try 구문 수행하기 전 c -->" + c);
005:     try {
006:       System.out.println("try 구문으로 들어옴");
007:       System.out.println("나누기 연산하기 전 try 구문");
008:
009:       c = a / b02;  //(1)
010:       System.out.println(" (1) c -> " + c);
011:
012:       c = a / b01; //(2)
013:       System.out.println(" (2) c -> " + c);
014:
015:       c = a / b02; //(3)
016:       System.out.println(" (3) c -> " + c);
017:     }
```

<예제> 하나의 try문과 여러 개의 catch 블록

```
018:    //하위 클래스로 특수한 사항에 대한 예외를 처리하는 클래스를 우선적으로 기술
019:    catch(ArithmeticException e) {
020:      System.out.println("예외가 발생하여 ArithmeticException객체가 잡음->" + e);
021:    }
022:    //반면 위에 기술된 catch 구문에서 처리하지 못한 예외만 Exception 객체가 처리
023:    catch(Exception e) {
024:      System.out.println("예외가 발생하여 Exception 객체가 잡음->" + e);
025:    }
026:    System.out.println("try 구문을 수행한 후 c-->" + c);
027:  }
028:}
```


finally

try/catch 구문에서 예외가 발생하거나 발생하지 않더라도 반드시 실행해야 하는 문장들이 있을 때, 바로 finally 블록 내에 기술

<예제> finally 구문

```
001:public class ExcepTest05 {
002:  public static void main(String[] args) {
003:    int a=10, b01=0, b02=2, c=10;
004:    System.out.println("try 구문 수행하기 전 c -->" + c);
005:    try {
006:      System.out.println("try 구문으로 들어옴");
007:      System.out.println("나누기 연산하기 전 try 구문");
008:
009:      c = a / b02;  //(1)
010:      System.out.println(" (1) c -> " + c);
```

finally

```
011:
012:     c = a / b01; //(2)
013:     System.out.println(" (2) c -> " + c);
014:
015:     c = a / b02; //(3)
016:     System.out.println(" (3) c -> " + c);
017:   }
018:   catch(ArithmeticException e) {
019:     System.out.println("예외가 발생하여 ArithmeticException객체가 잡음->" + e);
 020:   }
021:   catch(Exception e) {
022:     System.out.println("예외가 발생하여 Exception 객체가 잡음->" + e);
023:   }
024:   finally{
025:     System.out.println("try/catch 구문에 대한 정리 작업 c-->" + c);
026:   }//예외처리 끝
027:   System.out.println("try 구문을 수행한 후 c-->" + c);
028:  }
029:}
```

3. 다양한 종류의 예외

1. try 구문으로 진입
2. try 구문 안에서 예외가 발생하면
3. catch 구문을 순차적으로 살펴보면서 일치하는 예외가 있는지 조사하여 해당 블록으로 간다.
4. 해당 catch 블록을 실행하여 에러처리를 한다.
5. finally는 예외 발생 유무에 상관없이 무조건 실행한다.

NullPointerException

```
001:class MyDate{   //클래스의 초기값을 지정함
002:  int year=2022;
003:  int month=4;
004:  int day=1;
005:}
006:public class ExcepTest06{
007:  public static void main(String[] args) {
008:    //MyDate d;
009:    MyDate d=null;
010:    try{
011:      System.out.println(d.year+ "/" +d.month+ "/" +d.day);
012:    }catch(Exception e){
013:      System.out.println("예외 발생->" + e);
014:      d=new MyDate();
015:      System.out.println(d.year+ "/" +d.month+ "/" +d.day);
016:    }
017:  }
018:}
```


ArrayIndexOutOfBoundsException

```
001:public class ExcepTest07{
002:  public static void main(String [] args){
003:    int num[]={1,2,3};
004:    try{
005:      System.out.println("Test-1");
006:      num[4] = 4;
007:      System.out.println("Test-2");
008:    }catch(Exception e){
009:      System.out.println("예외발생 -> " + e);
010:    }
011:    System.out.println("Test-3");
012:  }
013:}
```


4. 예외의 인위적인 발생

throw 예외객체;
또는
throw new 예외클래스(전달인자);

<예제> 예외가 발생하지 않는 프로그램 [파일이름:ThrowTest.java]

```
001:public class ThrowTest{
002:  public static void exp(int ptr){
003:
004:  }
005:  public static void main(String[] args) {
006:    exp(0);
007:  }
008:}
```


<예제> 인위적으로 예외 발생하기

```
001:public class ThrowTest02{
002:  public static void exp(int ptr){
003:    if(ptr == 0)
004:      throw new NullPointerException();  //프로그래머가 예외 발생시
005:    }
006:    public static void main(String[] args) {
007:      exp(0);
008:  }
009:}
```


5. 호출 메서드에 예외 전달

```
static void method() throws UserError{
            throw new UserError();
}
```

★ throw 와 throws 키워드

자바 예외 처리 구문에서 throw 와 throws 키워드는 단어가 비슷해 사용할 때 용법에 주
해야 합니다.

① throw 키워드

throw 는 예외를 일부러 발생시킬 때 사용하는 키워드입니다.

▶ throw new 예외 클래스 생성자 ; 형식으로 사용합니다.

②.throws 키워드

throws 는 발생된 예외를 자신이 직접 처리하는 것이 아니라 자신을 호출한 객체로 떠넘
기는 역할을 하는 키워드입니다.

▶ public void 메서드이름(매개변수) throws Exception { ... } 형식으로 사용합니다.

6. 사용자 정의 예외

```
001:class UserException extends Exception {
002:  public UserException(String str){//새롭게 정의한 예외 클래스의 생성자에
003:    super(str); //넘겨진 값을 슈퍼 클래스인 Exception의 생성자에게 넘겨줌
004:  }
005:}
006:class ExcepTest09 {
007:  public static void main(String [] args){
008:   try{
009:     int a = -11 ;
010:     if( a <= 0){
011:        //사용자가 정의한 예외를 인위적으로 발생시킴
012:       throw new UserException("양수가 아닙니다.");
013:     }
014:   }
015:   catch(UserException e){
016:     System.out.println( e.getMessage());
017:   }
018:  }
019:}
```