예외처리
(Exception)

예외(Exception)

Exception
예외처리
: 예기치 않았던 에러가 발생하여 프로그램이 중간에 멈추는 것을 미리 예방하는 것

에러 : 중대한 오류 (처리 할 수 없는 영역)
오류 : 미미한 오류 (처리 할 수 있는 영역)
의해 프로그램이 비정상적인 종료가 되는 것을 막겠다.(예외처리 목적)
오류(예외)

예외 – Runtime Exception(unchecked Exception)
- Exception (checked Exception)

– unchecked Exception (예외처리 선택) :프로그래머의 실수에 의해서 생기는 예외
- checked Exception (예외처리 필수):사용자의 실수와 같은 외적인 요인에 의해 발생하는 예외

Checked Exception
- 예외처리방식 – 내가직접 try~ catch()
- 던지는 방법 throws Exception (예외를 미루는 방법)

Unchecked Exception
- try~catch()

예방처리 (Exception handling) 방법

1. try~catch 이용하기

- try{
  data =System.in.read( );
  }
  catch(IOException e)
  {
  System.out.println(e.getMessage());
  }
- try~ catch ~ catch
- try~catch ~ catch ~finally

2 . throws (선택예외의 경우 사용 못함)

예외를 내가 처리하지 않고 넘길때는 throws

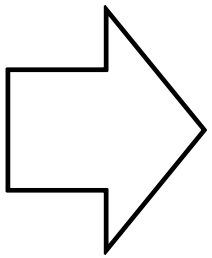
주의사항:

try ~catch ~ catch 로 작업할 때 큰 범위의 Exception이 먼저 나오면 안된다

예외처리

예기치 않은 상황에 발생되었을 때 예외를 일으킴
(예외는 미리 정의되어 있어야 함)
기본적으로 정의된 예외가 있고
사용자가 예외를 만들 수도 있음

예외를 일으키는 즉..
(예외를 발생시키는 코드를 만나야
예외가 발생함)



예외발생시 어떻게 할 것인가에
대한 대비코드를 작성하는 것

Integer.parseInt() 메서드로 예외 만나기

```
public  class MainTest
{
    public static void main(String[] args)
   {
      System.out.println( args[0] );
      System.out.println( args[1]);

      int su1=0, su2=0, sum=0;


      su1 = Integer.parseInt(args[0]);            //   "5"
      su2 = Integer.parseInt(args[1]);           //   "10"


      sum =su1+su2;

      System.out.println(  "합계=  "+sum);
   }
}
```

예외만나기

```
public  class MainTest
{
    public static void main(String[] args)
   {
      System.out.println( args[0] );
      System.out.println( args[1]);
      int su1=0, su2=0, sum=0;

     try{
         su1 = Integer.parseInt(args[0]);          // “5”
         su2 = Integer.parseInt(args[1]);          // “10”

      }catch( Exception e)
      {
            System.out.println( e.toString() );
            System.out.println(“숫자로 입력하세요”);
      }
      sum =su1+su2;
      System.out.println(“합계=“+sum);
   }
}
```

예외만나기

```
public class ExceptionEx
{
	public static void main( String[] args)
	 {
		Scanner sc = new Scanner(System.in);
		int result;

		System.out.print(  "나뉨수를 입력:"  );
		int divided = sc.nextInt();

		System.out.print(  "나눗수를 입력:"  );
		int divisor = sc.nextInt();

		result =divided /divisor;
		System.out.println(  "결과 :"    + result);

	 }
}
```

```java
package ioEx;


import java.io.IOException;


public class Ex01 {
		public static void main(String[] args) {
		//한바이트로 읽어옴
		try {
			int su = System.in.read();
			System.out.println( (char) su);

			su= System.in.read();
			System.out.println( (char) su);

		} catch (IOException e) {
		   e.printStackTrace();
		}
	}
}
```

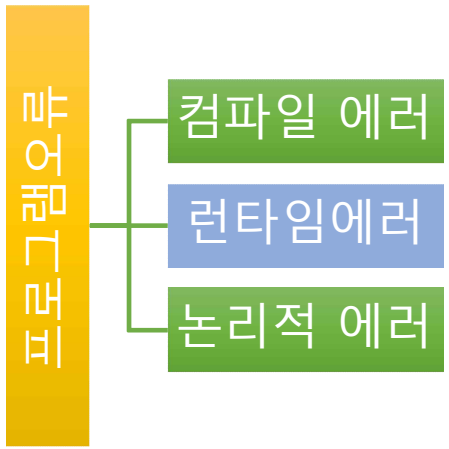
논리적에러 : 작성의도와 다르게 동작

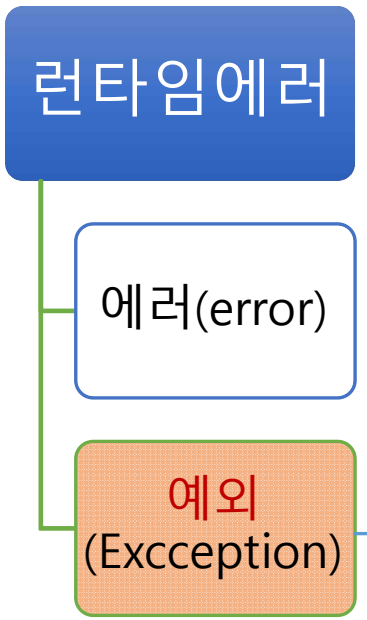
컴파일러 에러:
자바컴파일러
구문체크 (문법체크)
번역
최적화

실행 에러: (java런타임 에러)
JVM에 의해 실행될 때 문제가 발생하는 것

에러 (error) 프로그램 코드에 의해서 수습될 수 없는 심각한 오류 (out of memory)
예외 (exception) 프로그램 코드에 의해서 수습될 수 있는 다소 미약한 오류
프로그램이 죽지 않고 잘 돌아 가게 하기 위함

- 배열의 범위를 벗어 날 때 (선택)
- 숫자변환에 문제가 있을 때 (선택)
- 읽을 파일이 없을 때 (필수)
- 네트워크에 문제가 있을 때 (필수)

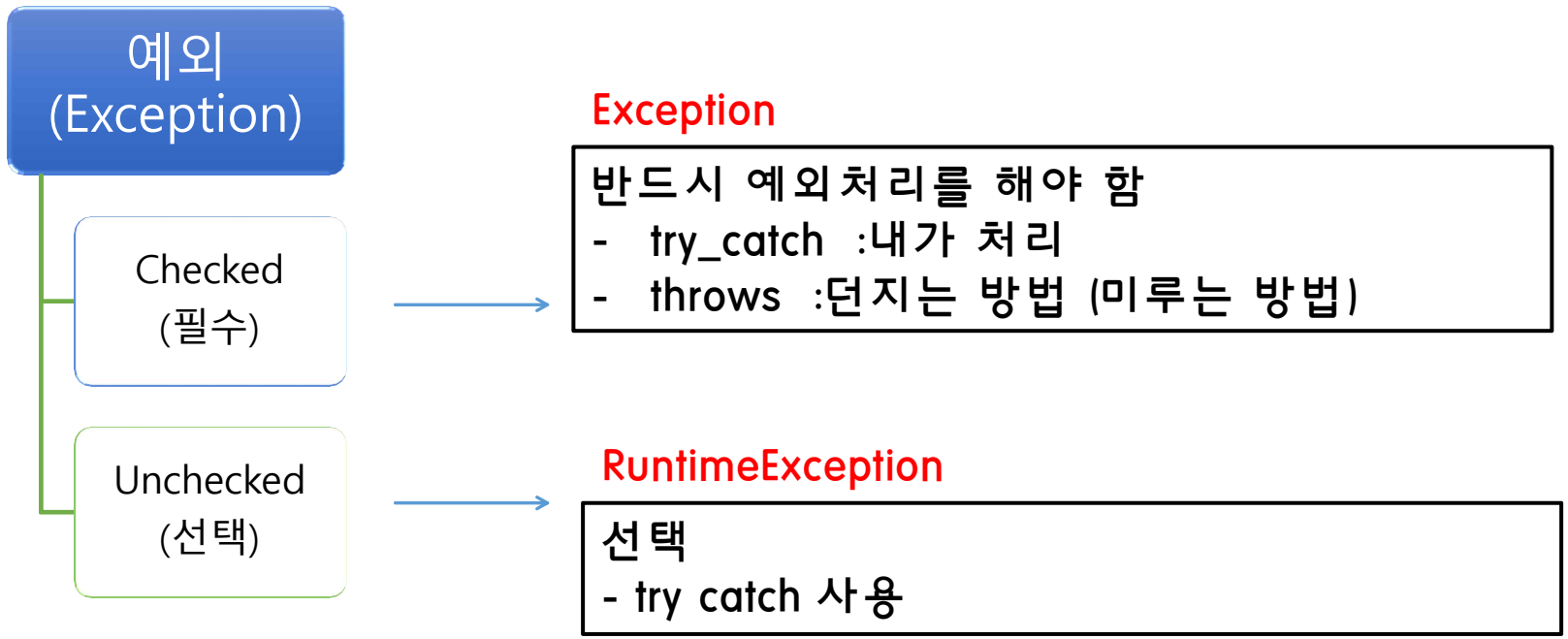




예외 처리

예외 처리 (Exception handling)

- **정의: 예외의 발생에 대비한 코드를 작성하는 것**
- **목적: 프로그램의 비정상적인 종료를 막고 정상적인 상태유지**


예외
(Exception)
Checked
(필수)
Unchecked
(선택)
Exception
반드시 예외처리를 해야 함
- try_catch :내가 처리
- throws :던지는 방법 (미루는 방법)
RuntimeException
선택
- try catch 사용

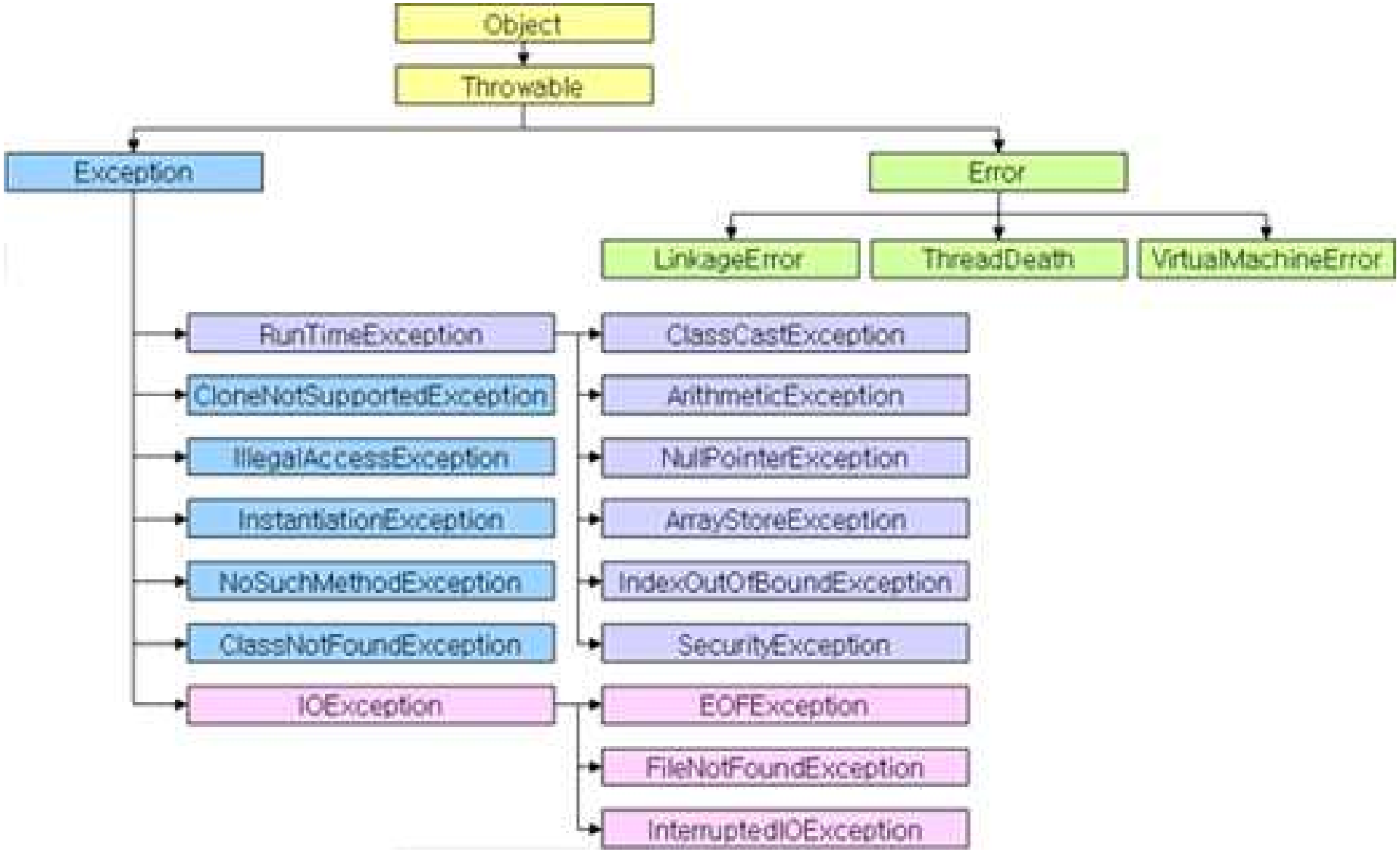
```
//필수이냐 선택이냐 예외가  에 따라
public class MyCalculator {

      //나누기 기능
       public  double  divide( int num1, int num2){
              if( num2 ==0) {
                     throw new ArithmeticException();
              }
              return num1/ num2;

     }

    // 더하기 기능
       public int add( int num1, int num2)  throws MyException {
              if (num1 <0 ll  num2 <0) {
                     throw new Exception("0보다 큰값이어야 합니다.");
              }

              return num1+ num2;
       }


}
```

```
public class MyCalculatorTest {

	public static void main(String[] args) {

	MyCalculator m = new MyCalculator();
	m.divide(10, 0);    //선택예외이기 때문에 예외처리가 반드시 해야 할 필요없음

	//예외가 발생하면 예외가 터지고 프로그램은 비정상적인 종료함


	int result=0;

	try {
	  result  = m.add(-7, 0);    // 필수예외이므로 반드시 예외처리를 해야함
	} catch (Exception e) {

	  e.printStackTrace();
	}

	//예외처리를 해 주면 예외상황이 발생되었을 때 대비한 코드가 실행되고 프로그램은 정
	상적으로 유지 된다.

	 System.out.println( " 프로그램 정상종료 "  + result);
	}

}
```

```
public class Ex01 {

	public static void main(String[] args) {

	  int[] arr= new int[5];

	  arr[5]=90;

	  System.out.println( "프로그램 정상 종료" );
	}
}
```

```
import java.io.*;

public class Ex02 {

   public static void main(String[] args) {

         int[] arr= new int[5];

         try {
               arr[5]=90;
         }catch(ArrayIndexOutOfBoundsException e)
         {
            System.out.println("ArrayException==>" +  e.getMessage());
            System.out.println("배열첨자 에러 : 실행시 데이타를 입력하세요");
         }

         System.out.println( "프로그램 정상 종료" );

        }
}
```

```
public class TryCatchCatchFinallyTest {

        public static void main(String[] args) {
           int data=0;

          try {
                System.out.println("실행시 넘어온 인자는=> args[0]=" +args[1]);
                data=Integer.parseInt(args[0]);
          }catch(ArrayIndexOutOfBoundsException e)
          {
              System.out.println(" 배열첨자 에러가 발생하면  이 코드가  실행됨 : " +  e.getMessage());
              System.out.println(" 실행시 데이타를 입력하세요");
          }catch(NumberFormatException e)
          {
               System.out.println("숫자변환 예외가 발생하면 이 코드가 실행됨  :  " + e.getMessage());
               System.out.println("숫자형태로 입력해라");
          }
          finally {
          System.out.println("항상 실행됨 : 예외상황이든 아니든 무조건 실행하는 부분이 있다면 여기에...");
          }

           System.out.println(  " 프로그램 정상종료 : 정수  ==➔ "   +data);
          }
}
```


Object
Throwable
Exception
Error
LinkageError
ThreadDeath
VirtualMachineError
RunTimeException
CloneNotSupportedException
IllegalAccessException
InstantiationException
NoSuchMethodException
ClassNotFoundException
IOException
ClassCastException
ArithmeticException
NullPointerException
ArrayStoreException
IndexOutOfBoundException
SecurityException
EOFException
FileNotFoundException
InterruptedIOException

사용자예외만들기

```
public class 음수예외필수 extends Exception {
        public 음수예외필수() {
                super("음수안됨 !!");
        }
}
```

```
public class 음수예외선택 extends RuntimeException {
        Public 음수예외선택() {
                super("음수안됨 !!");
        }
}
```

```
public class MyCalculator {
	public int add( int num1, int num2) throws 음수예외 {
			if( num1< 0 || num2 <0) {

				throw new 음수예외필수( );
			}
			return num1+ num2;
			}


			public int add2( int num1, int num2) {
				if( num1< 0 || num2 <0) {

				throw new 음수예외선택();
			}
			return num1+ num2;
	}


}
```