Chapter 10 예외 처리

01 예외와 기본 예외 처리

* 예외 : 프로그램 실행 중 프로그램이 중단되는 오류
* 예외처리 : 오류를 대처할 수 있게 하는 것
* 에러 : 프로그램이 컴파일조차 안 되게 하는 프로그래밍 언어의 문법적 오류

🡺indexOutOfRangeException 기본 예외 처리

1. 예외 상황 확인하기
2. 기본 예외 처리하기

* 배열 길이 확인해서 입력 값이 배열 길이 넘으면 잘못되었다고 알려줌

02 고급 예외 처리

* Try 키워드, catch 키워드, finally 키워드로 예외를 처리하는 방법
* 고급 예외 처리 형식(try catch finally 구문)

Try{ //예외가 발생하면 }

Catch(Exception exception){ //여기에서 처리한다}

Finally{ // 여기는 무조건 처리한다 }

* Catch 구문이나 finally 구문을 안 쓰는 경우도 있을 수 있다.

1. 예외 상황을 파악하고
2. 고급 예외 처리를 한다. Try catch finally

* Finally 구문
  + 만약에 오류가 발생해서 catch 문에서 처리를 했는데 finally 구문이 없으면 catch 문 다음에 실행하는 문장은 실행되지 않는다.
  + Catch 구문에서 return 키워드나 break 을 만나도 finally 에 있는 내용들은 실행이 된다.
* Finally 구문과 return 키워드
* Finally 구문 내부는 무조건 실행하고 끝낸다는 규칙이 있어서 중간에 구문을 벗어나는 키워드는 사용 불가능 예를 들면 return

03 예외 객체

* 예외 발생 시 어떤 예외가 발생했는지와 관련된 정보를 함께 전달해주는 것
* 예를 들면

Try{ } catch (Exception exception){ } // Exception 클래스의 인스턴스로 예외 객체라고 부른다.

* 예외 객체에서 정보를 뽑아 낼 수 있는데 GetType 으로 타입과 Message 와 StackTrace 등의 정보를 뽑아 낼 수 있다.

04 예외 객체를 사용한 예외 구분

1. 예외 상황 확인하기
2. 예외 상황의 종류에 따라서 다른 작업을 실행하게 할 수 있다.

* FormatException exception 이런식으로 인자를 처리하면 자료형의 문제이고
* IndexOutOfRangeException exception 이런식으로 인자를 처리하면 인덱스의 문제이다.
* 이런식으로 타입을 나누어서 예외 상황을 처리할 수 있다.
* Catch 구문에는 var 키워드를 사용 할 수 없다.
* Try{ } catch (var exception) { } 이런식으로 사용 불가!

05 예외 강제 발생

* 예외 강제 발생 방법

Throw new Exception( );

1. 예외 던지기 확인하기

Throw new Exception( );

1. 강제로 던진 예외의 예외 처리하기

Try { throw new Exception(); }

Catch ( Exception exception ) { Console.WriteLine(“예외가 발생했습니다.”); }

* 사용자 정의 예외

Class CustomException : Exception {

Public CustomException(string message) : base(message){

}

}

Class Program{

Static void Main(string[] args)

{

Try{ throw new CustomException(“사용자 정의 예외”);

Catch( CustomException exception) {

Console.WriteLine(exception.Message);

}

}

}