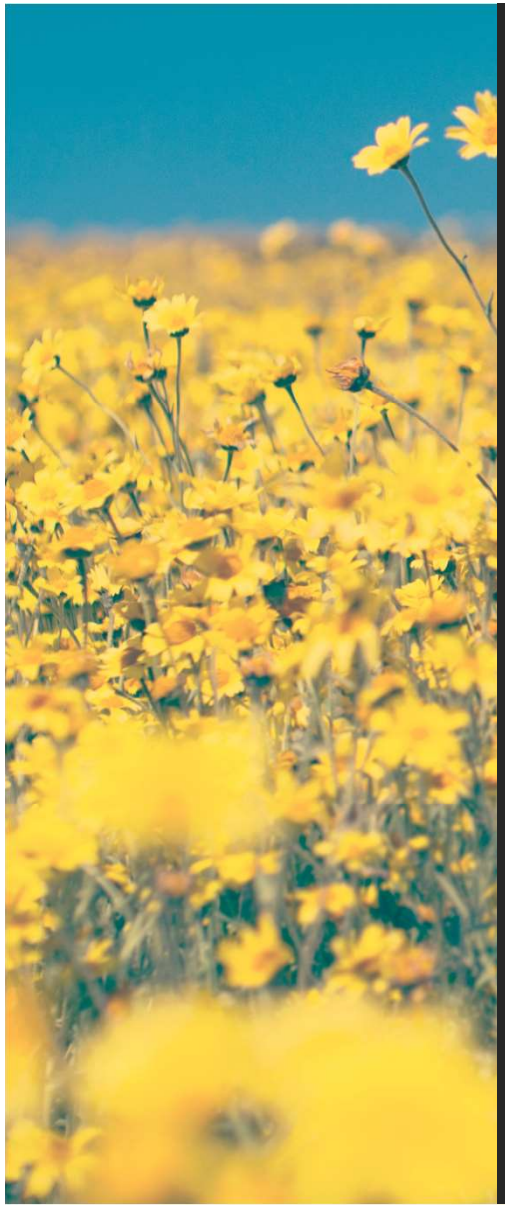


JAVA

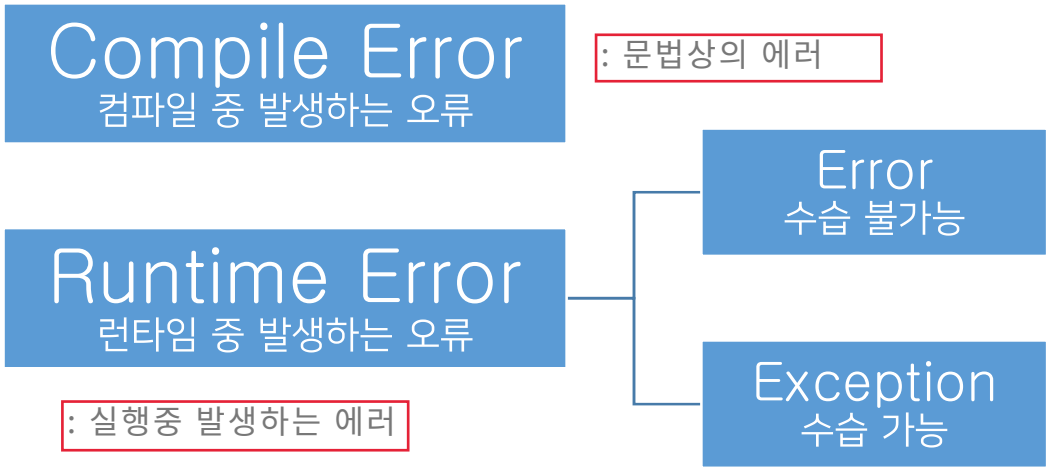
GU, DA HAE


에 러
t r y
c a t c h
t h r o w
t h r o w s
예 외 클 래 스
JAVA
예 외 처 리

JAVA

오류

Exception Handling



JAVA

예외 처리

Exception Handling

예외(exception)는 프로그램의 실행 도중에 발생하는 '수습 가능한 에러'를 또는 프로그래머가 예상하지 못한, 프로그램의 논리에 맞지 않는 상황을 뜻한다.

[예]

나이를 입력 하는데, 0보다 작은 값이 입력된다.

나눗셈을 위해 두 개의 정수를 입력 받는데, 제수(나누는 수)로 0이 입력된다.

주민등록번호를 숫자로만 13자리만 입력하라고 했더니, 중간에 -를 포함하여 14자리를 입력된다.

보통 위와 같은 상황은 if문으로 처리한다. 그러나 if문은 예외처리 이외의 용도로도 사용되기 때문에, 프로그램 코드 상에서 해당 if문이 예외처리 부분인지 구분하기 어렵다.

JAVA

예제

```
import java.util.Scanner;

public class Ex01{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);

        System.out.println("나누기를 시작합니다.");
        System.out.print("첫 번째 수 입력 : ");
        int a = sc.nextInt();

        System.out.print("두 번째 수 입력 : ");
        int b = sc.nextInt();

        if(b == 0)
            System.out.println("제수(나누는 수)는
0이 될 수 업습니다.");
        else {
            System.out.println("몫 : " + a / b);
            System.out.println("나머지 : " + a%b);
        }
    }
}
```

JAVA

try, catch

Exception Handling

예외처리란 **프로그램의 비정상 종료를 막고, 정상적인 실행상태를 유지하는 것**을 말한다. 자바에서는 예외처리를 위해 try, catch 키워드를 제공한다.

```
try{
    예외가 발생할 만한 코드
}
catch(AAA e){
    예외처리를 위한 코드
}
```

try로 예외가 발생하는지 검사하고, try에서 AAA형의 예외가 발생하면 **catch로 발생한 예외를 처리**한다.

발생하는 예외는 JVM이 알아서 감지한다.

JAVA

예제

```java
import java.util.Scanner;
public class Ex01{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("나누기를 시작합니다.");
        System.out.print("첫 번째 수 입력 : ");
        int a = sc.nextInt();
        System.out.print("두 번째 수 입력 : ");
        int b = sc.nextInt();
        try {
            System.out.println("몫 : " + a / b);
            System.out.println("나머지 : " + a % b);
        }catch(ArithmeticException e) {
            System.out.println("나누기를 할 수 없습니다.");
        }
     }
}
```

JAVA

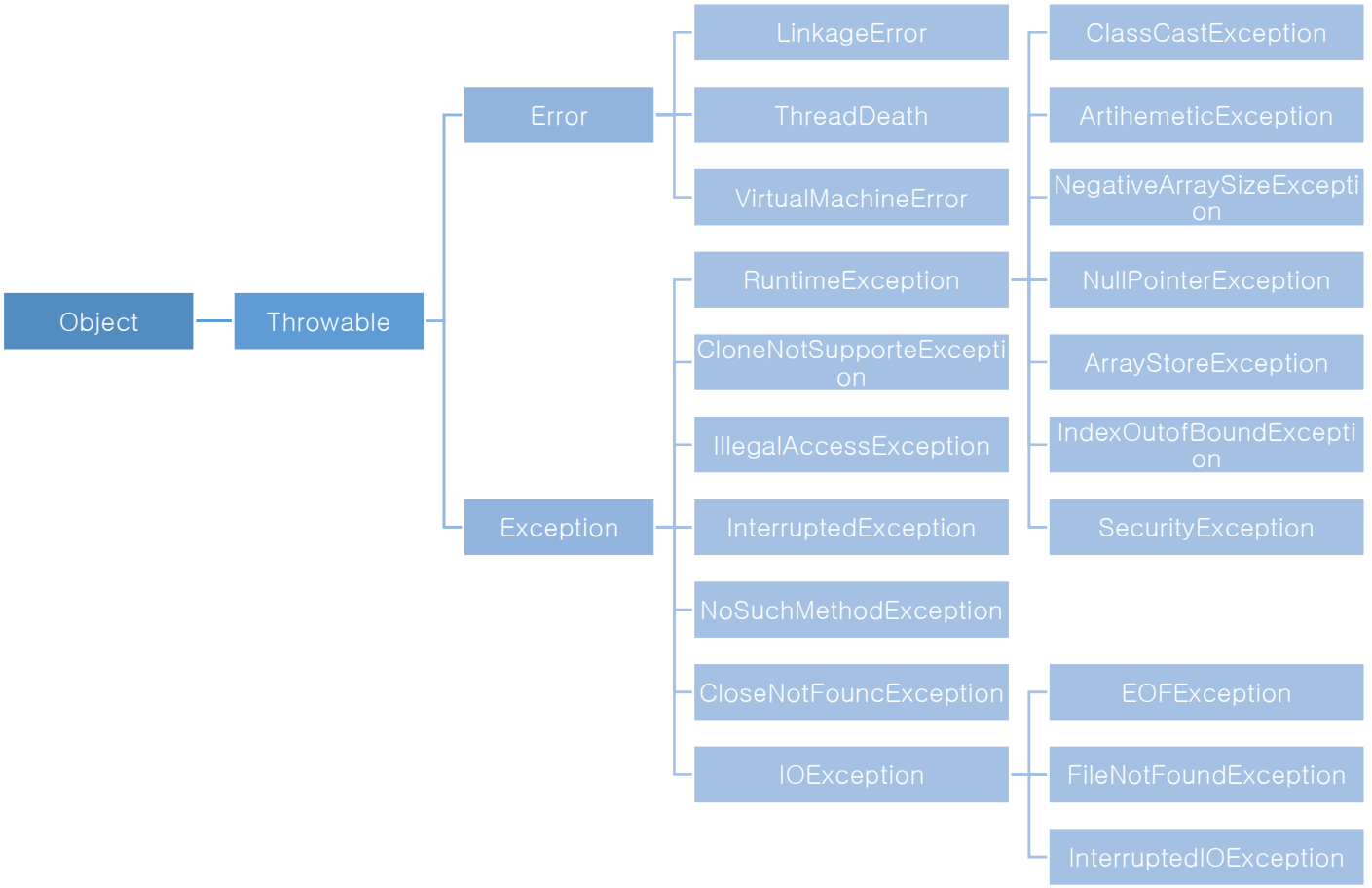
예제

```java
import java.util.Scanner;


public class Test {
    public static void main(String[] args) {
        int[] result = new int[3];

        try {
            Scanner sc = new Scanner(System.in);
            System.out.println("나누기 횟수 : ");
            int size = sc.nextInt();

            for(int i = 0; i< size ; i++) {
                System.out.println("첫 번째 숫자 : ");
                int a = sc.nextInt();
                System.out.println("두 번째 숫자 : ");

                int b = sc.nextInt();


                result[i] = (int)(a/b);
            }
        }catch(ArithmeticException e) {
            System.out.println("나누기 에러");
            e.printStackTrace(); // 에러 발생 이유, 경로 출력
        }catch(IndexOutOfBoundsException e) {
            System.out.println("인덱스 범위 에러");
        }
    }
}
```

JAVA

try, catch

Exception Handling



JAVA

해보기

up, down 게임 프로그램을 실행하는 도중 숫자가 아닌 문자가 입력될 수도 있다. 이때 예외처리를 통해 이 상황을 해결해 보자. 아래는 해당 프로그램의 실행 내용이다.

```
[up or down]
Call Number : 23
-> up
Call Number : 60
-> down
Call Number : 50
-> down
Call Number : 40
-> up
Call Number : 43
-> Congratulations!
```

java.util.InputMismatcheException

java.util.radnom
nextInt() : int형 범위
nextInt(n) : 0~n 사이의 숫자를 무작위로 생성

sc.nextLine() // 버퍼 비우기

JAVA

finally

Exception Handling

예외 발생과 상관없이 항상 실행되어야 하는 코드들은 finally 로 묶는다.

```
try{
    예외가 발생할 만한 코드
}
catch(AAA e){
    예외처리를 위한 코드
}finally{
    반드시 실행해야 할 코드
}
```

finally는 예외가 발생하지 하지 않든 반드시 실행된다.

JAVA

예제

```java
import java.util.Scanner;

public class Ex01{
	public static void main(String[] args){
		Scanner sc = new Scanner(System.in);

		System.out.println("나누기를 시작합니다.");
		System.out.print("첫 번째 수 입력 : ");
		int a = sc.nextInt();

		System.out.print("두 번째 수 입력 : ");
		int b = sc.nextInt();


		try {
			System.out.println("몫 : " + a / b);
			System.out.println("나머지 : " + a % b);
		}catch(Exception e) {
			System.out.println("나누기를 할 수 없습니다.");
		}finally {
			System.out.println("FINALLY!!");
		}
	}
}
```

JAVA

예외 클래스

Exception Handling

Exception 클래스를 상속하여 프로그램에 필요한 예외를 직접 처리할 수 있다.

```
class TestException extends Exception{
    ...
}
```

```
try{
    throw new TestException();
}catch( TestException e){
    ...
}
```

새로운 예외클래스를 정의 한 뒤, throw 키워드로 예외가 발생했을 시 새로운 예외 인스턴스를 생성한다.

JAVA

예제

```
import java.util.Scanner;

class InputAgeException extends Exception{

    public InputAgeException(){

        super("정확한 나이를 입력해주세요.");

    }

}

 // 상위 클래스의 생성자에 문자열을 지정하면, getMessage() 메서드를 호출했을 때 해당 문자열이 반환된다. 예외가 발생한 이유를 주로 표시한다.
```

```
public class Ex01{

      public static void main(String[] args){

          System.out.println("나이 입력 : ");


          try {

              int age = new Scanner(System.in).nextInt();

              // 예외가 발생해야 할 상황인지 검사

              if(age < 0 || age > 120)

                  throw new InputAgeException(); // 예외 발생


              System.out.println("입력하신 나이는 " + age + “세입니다.");

          }catch(InputAgeException e) {

              System.out.println(e.getMessage());

          }

      }

}
```

JAVA

예외 처리 순서 Exception Handling

예외가 발생한 메서드에서 처리하지 않으면, 호출한 곳으로 예외 처리를 넘긴다. 이때, 반드시 이 사실을 명시해야 한다.

```
class Test throws SomeException{
        ...
        throw new SomeException;
        ...
}
```

메서드 내부에서 예외를 처리하지 않으면 호출한 메서드로 예외를 떠넘긴다.

main 메서드에서 예외를 처리하지 않으면 아래와 같은 순서로 JVM이 예외를 처리한다.

1. getStackTract() 메서드 호출
2. 프로그램 종료

JAVA

예제

```java
import java.util.Scanner;

class InputAgeException extends Exception{
    public InputAgeException(){
        super("정확한 나이를 입력해주세요.");
    }
}

public class Ex01{
    public static int inputAge() throws InputAgeException{
        int age = new Scanner(System.in).nextInt();
        if(age < 0 || age > 120)
            throw new InputAgeException(); // 예외 발생
        return age;
    }
    public static void main(String[] args){
        System.out.println("나이 입력 : ");

        try {
            int age = inputAge();
            System.out.println("입력하신 나이는 " + age + "입니다.");
        }catch(InputAgeException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

JAVA

예제

```java
import java.util.Scanner;

class InputAgeException extends Exception{
    public InputAgeException(){
        super("정확한 나이를 입력해주세요.");
    }
}

public class Ex01{
    public static int inputAge() throws InputAgeException{
        int age = new Scanner(System.in).nextInt();
        if(age < 0 || age > 120)
            throw new InputAgeException(); // 예외 발생
        return age;
    }
    public static void main(String[] args) throws InputAgeException{
        System.out.println("나이 입력 : ");

            int age = inputAge();
            System.out.println("입력하신 나이는 " + age + "입니다.");

    }
}
```

JAVA

해보기

지금까지 배웠던 내용을 토대로, 간단한 은행 계좌 관리 프로젝트를 완성해보자.

[사용자 종류]

일반 사용자, 관리자

[필요한 기능]

사용자 등록 : 사용자의 개인 정보와 계좌 번호를 등록한다. 관리자일 경우 개인 정보와 관리 번호를 등록한다.

예금액 출력 : 계좌번호를 입력하여 해당 계좌의 예금액을 확인한다.

출금 : 계좌번호를 입력하여 해당 계좌에서 출금한다. 이때 잔액보다 많이 출금할 수 없다.

입금 : 계좌번호를 입력하여 해당 계좌에 입금하다.

모든 사용자 정보 출력 : 관리자일 경우 모든 사용자의 정보를 볼 수 있다.