

# Object – Oriented Programming

---

LAB #9. Exception Handling & UML

# Exception

---

- Exception은 프로그램의 정상적인 수행 도중에 발생할 수 있는 오류를 나타냄
- Exception이 발생하면, 프로그램이 종료됨
- Exception handling을 통해서 종료되지 않고 실행이 유지되도록 할 수 있음
- 프로그램에서, 프로그래머는 exceptional case를 다루는 코드를 제공해야 한다.

# Not Catching Exceptions

---

```
Scanner keyboard = new Scanner(System.in);  
System.out.print("Enter age: ");  
int age = keyboard.nextInt();  
System.out.println("Your age is " + age);
```

사용자가 10 대신 'Ten' 을 입력하면...

```
Exception in thread "main" java.util.InputMismatchException  
    at java.base/java.util.Scanner.throwFor(Scanner.java:939)  
    at java.base/java.util.Scanner.next(Scanner.java:1594)  
    at java.base/java.util.Scanner.nextInt(Scanner.java:2258)  
    at java.base/java.util.Scanner.nextInt(Scanner.java:2212)  
    at practice09/practice09.test01.main(test01.java:9)
```

# Not Catching Exceptions

---

- 위의 상황에서 두 가지 일이 발생한다.
  - Java가 InputMismatchException을 발생시킴
  - 프로그램이 exception catch를 실패하여 충돌이 일어남
- try-catch 문으로 이 상황을 해결할 수 있다.

# try-throw-catch Basic

---

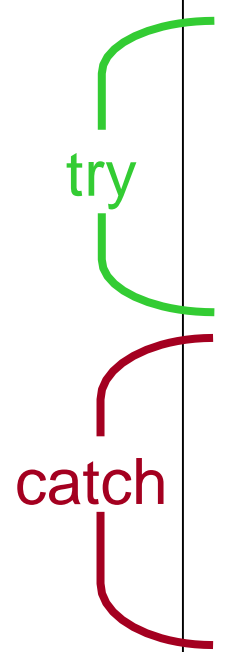
```
try {  
  
    // some code to attempt  
    // this code may throw an exception  
  
} catch (Exception e){  
  
    // catch the exception if it is thrown  
    // do whatever you want with it.  
  
}
```

# Catching an Exception

```
System.out.print("Enter age: ");

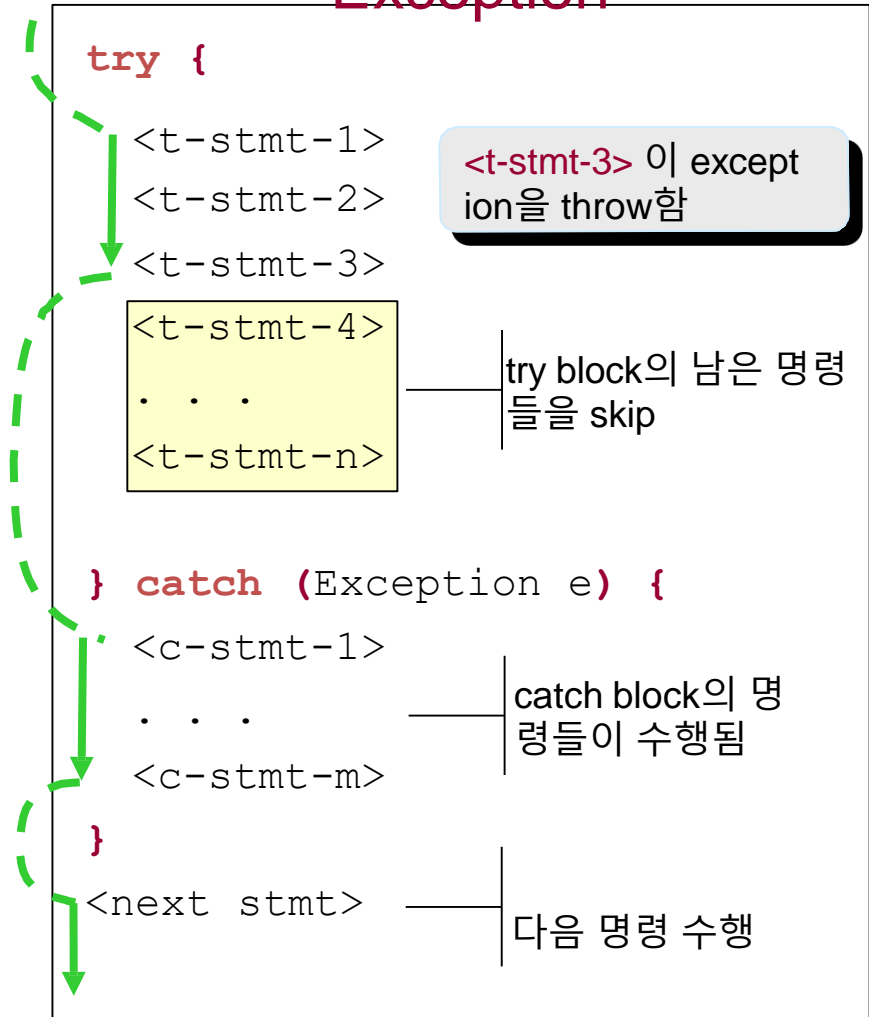
try {
    age = scanner.nextInt();
    System.out.println("Your age is " + age);
} catch (InputMismatchException e) {

    System.out.println("Invalid Entry. "
        + "Please enter digits only");
}
```

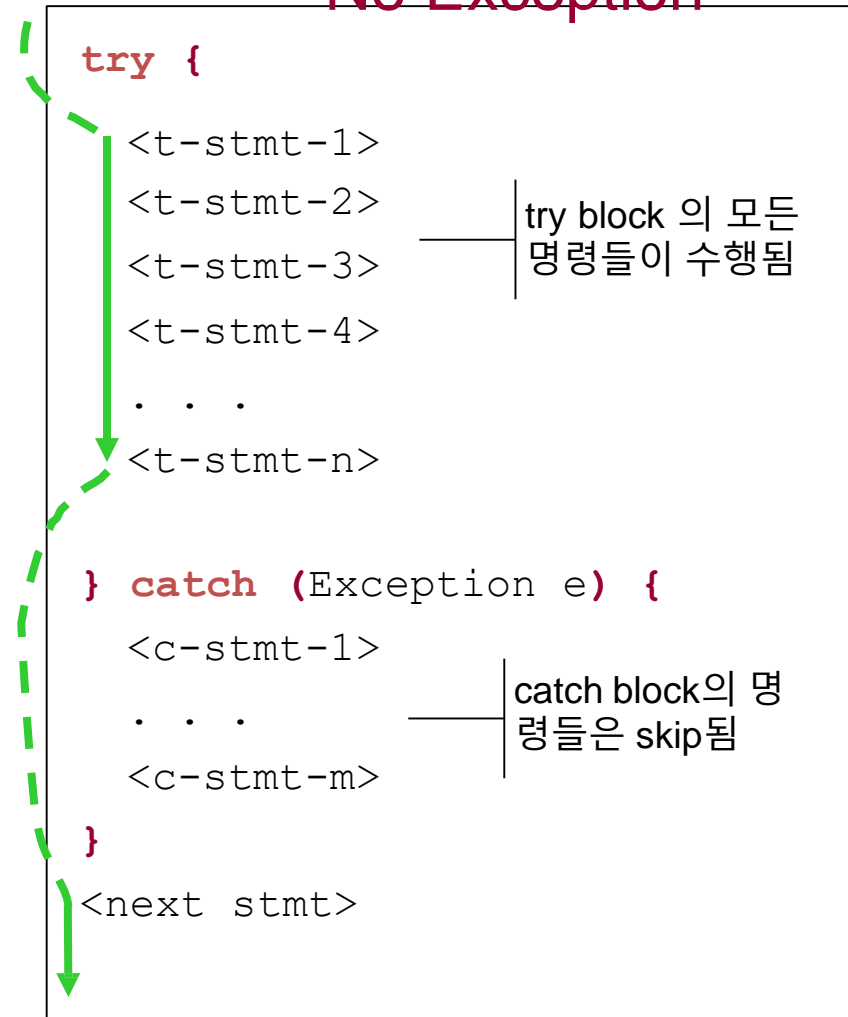
A diagram illustrating the structure of a try-catch block. A green curly brace on the left side groups the lines within the try block, with the word "try" written in green next to it. A red curly brace on the left side groups the lines within the catch block, with the word "catch" written in red next to it.

# try-catch Control Flow

## Exception



## No Exception



# try-catch

---

```
public static void main(String[] args) {  
    try {  
        System.out.println(1);  
        System.out.println(2);  
        System.out.println(0/0);  
        System.out.println(3);  
        System.out.println(4);  
    } catch (Exception e) {  
        System.out.println(5);  
        System.out.println(e.getMessage());  
    }  
    System.out.println(6);  
}
```

1  
2  
3  
4  
6

1  
2  
5  
/ by zero  
6



# Exception Object

---

- Exception 객체에 관하여 가장 중요한 2가지
  - exception의 type (i.e., exception class)
  - exception이 가지고 있는 message
- message는 instance variable로 exception 객체와 함께 전달된다.
- 이는 accessor 메소드인 getMessage()를 통해 불러올 수 있다.
  - > catch block에서 이 message를 이용 가능

# Exception Class

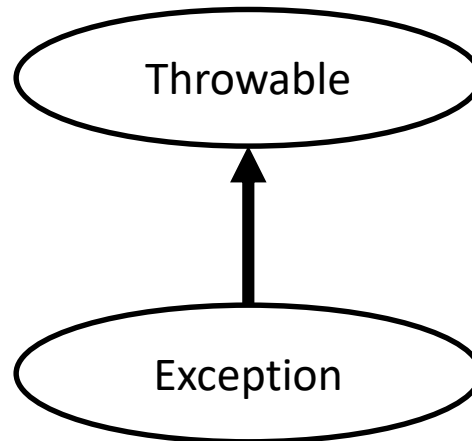
---

- Java의 표준 패키지에는 사전에 정의된 수많은 exception class들이 있다.
  - 예를 들면
    - IOException
    - NoSuchMethodException
    - FileNotFoundException
- 이러한 exception class들을 사용하기 위해서는 import 해야한다.
  - Import java.io.IOException

# Exception Message

---

- getMessage() Method
  - 모든 exception은 exception이 발생한 이유를 식별하는 message를 포함한 String instance variable을 가지고 있다.
  - getMessage() 메소드는 자세한 message 를 반환한다.



# Programmer-defined Exceptions

---

- Exception class는 프로그래머가 정의할 수 있으나, 이미 존재하는 Exception class로 부터 파생된 Class여야 한다.
- 다른 Exception class가 적합하지 않다면, 클래스 'Exception'을 base class로 사용할 수 있다.
- 적어도 2개 이상의 생성자가 정의되어야 한다.
- 해당 Exception class는 getMessage()를 상속 받는다.

# Exception Message type

---

- 다른 타입의 인자를 받는 Exception class 생성자를 정의 할 수 있다.
  - 이 생성자는 값을 instance variable에 저장한다.
  - 이 instance variable에 접근하기 위해 accessor 메소드를 정의해야 한다.

# Programmer-defined Exceptions

---

```
Public class MyException extends Exception{  
    // variables  
  
    public MyException(){  
        super("default message");  
        //perform other tasks  
    }  
  
    public MyException(String message){  
        super(message);  
        //perform other tasks  
    }  
  
    //other methods if needed  
}
```

# Example

---

**Display 9.5    An Exception Class with an int Message**

---

```
1  public class BadNumberException extends Exception
2  {
3      private int badNumber;

4      public BadNumberException(int number)
5      {
6          super("BadNumberException");
7          badNumber = number;
8      }

9      public BadNumberException()
10     {
11         super("BadNumberException");
12     }

13     public BadNumberException(String message)
14     {
15         super(message);
16     }

17     public int getBadNumber()
18     {
19         return badNumber;
20     }
21 }
```

# Multiple catch Blocks

---

- try-catch 문은 여러 개의 catch block을 가질 수 있다.

```
try {  
    ...  
    n = keyboard.nextInt();  
    ...  
} catch (InputMismatchException e) {  
    ...  
} catch (NumberFormatException e) {  
    ...  
}
```



# Multiple catch Blocks

---

- 여러 개의 catch 문을 사용할 때는 구체적인 exception을 먼저 catch해야 한다.

```
catch (BadNumberException e)
```

```
{ ... }
```

```
catch (Exception e)
```

```
{ ... }
```

- BadNumberException 이 Exception 보다 구체적이므로 먼저 catch되어야 한다.

# Throwing Exceptions

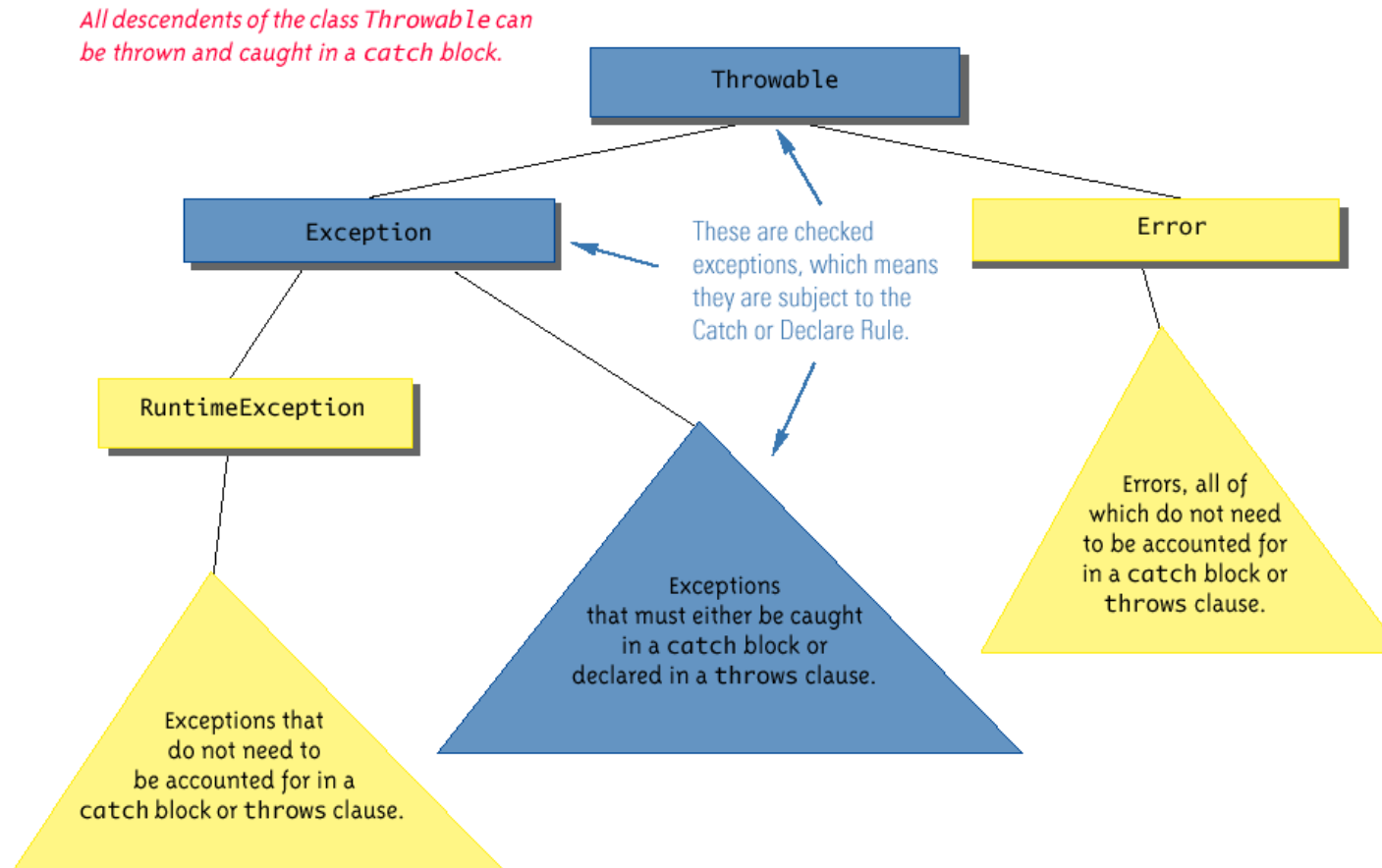
---

- method 내에서 exception을 handling 하지 않고 throw 할 수 있음
  - 해당 exception이 checked exception이라면 호출한 method에서 처리해주어야 함
  - 호출한 method에서도 throw 할 수 있지만, 결국에는 어디에서든 try-catch를 통해서 handling되어야 함

```
public static void isBadNumber(int num) throws BadNumberException {  
    if (num <= 0)  
        throw new BadNumberException(num);  
}
```

# Checked Exception vs Unchecked Exception

Display 9.10 Hierarchy of Throwable Objects



# Checked Exception vs Unchecked Exception

nextInt

...

Throws:

`InputMismatchException` - if the next token does not match the *Integer* regular expression, or is out of range  
`NoSuchElementException` - if input is exhausted  
`IllegalStateException` - if this scanner is closed

FileReader

...

Throws:

`FileNotFoundException` - if the named file does not exist, is a directory rather than a regular file, or

## Class InputMismatchException

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.lang.RuntimeException
        java.util.NoSuchElementException
          java.util.InputMismatchException
```

## Class FileNotFoundException

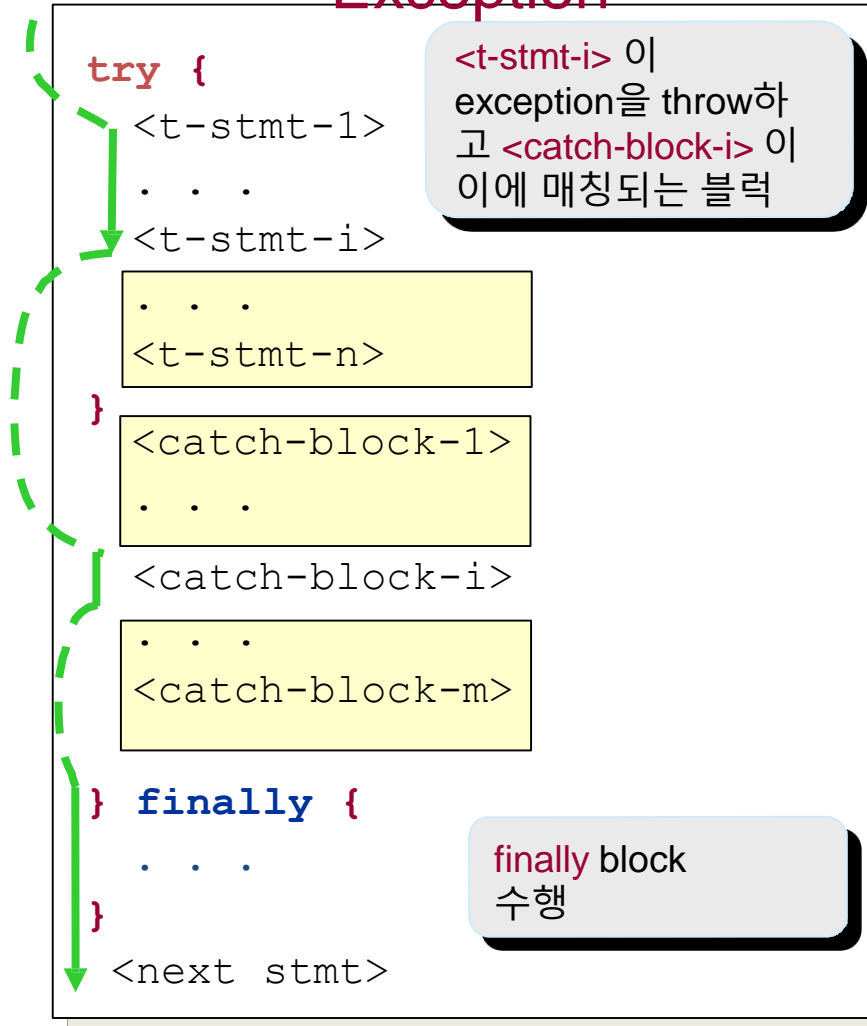
```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.io.IOException
        java.io.FileNotFoundException
```

```
int age = keyboard.nextInt();
```

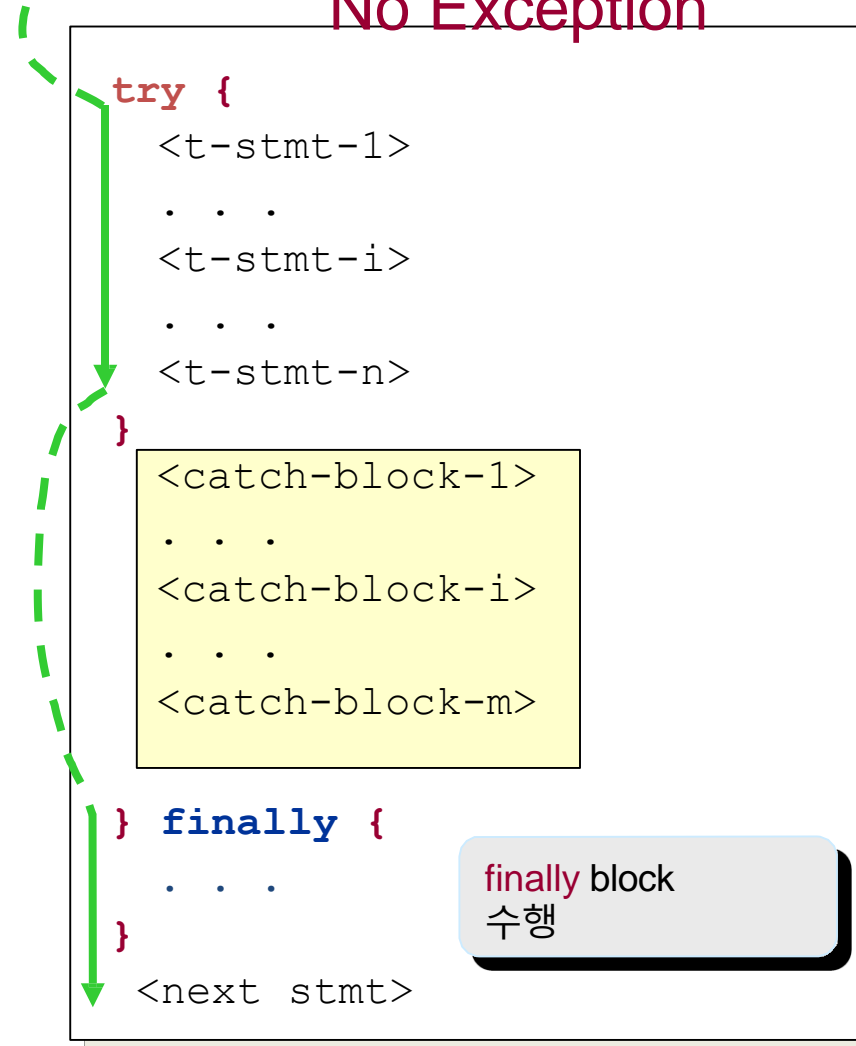
```
BufferedReader inputStream = new BufferedReader(new FileReader("morestuff2.txt"));
```

# The finally Block

## Exception

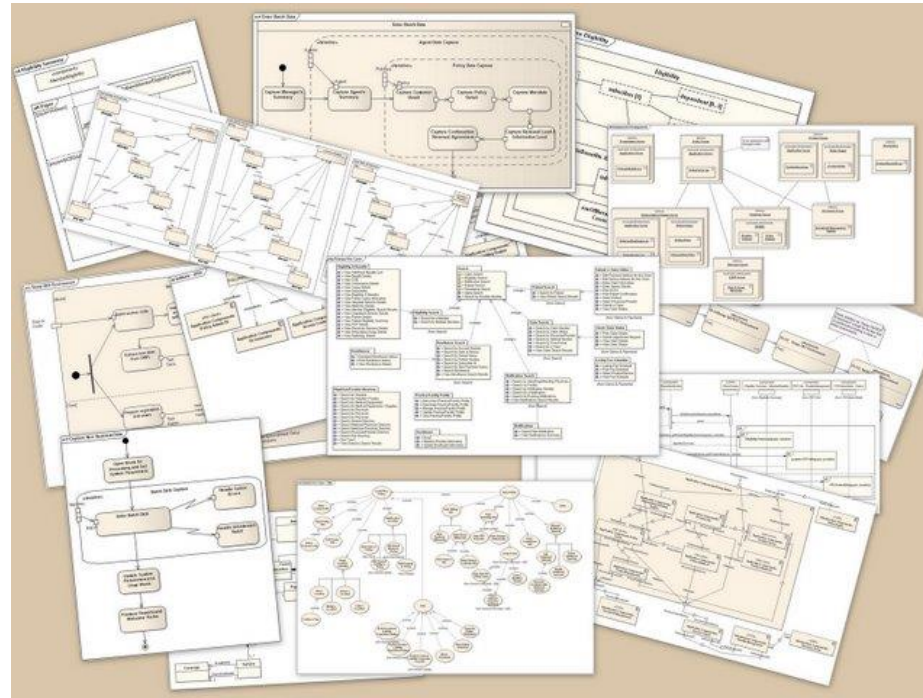


## No Exception



# UML

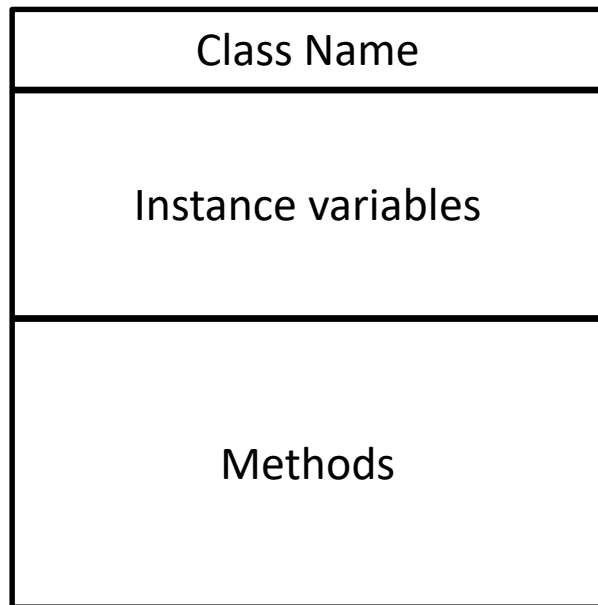
- UML(Unified Modeling Language, 통합 모델링 언어)  
: 객체 지향 프로그래밍 소프트웨어에서 설계, 문서화하기 위해 사용되는 그래픽 언어



# UML

---

## - Class Diagram



### **Instance variable:**

(modifier) (variable name): (type)

ex) private double side

→ - side: double

### **Method:**

(modifier) (method name)((parameters)): (return type)

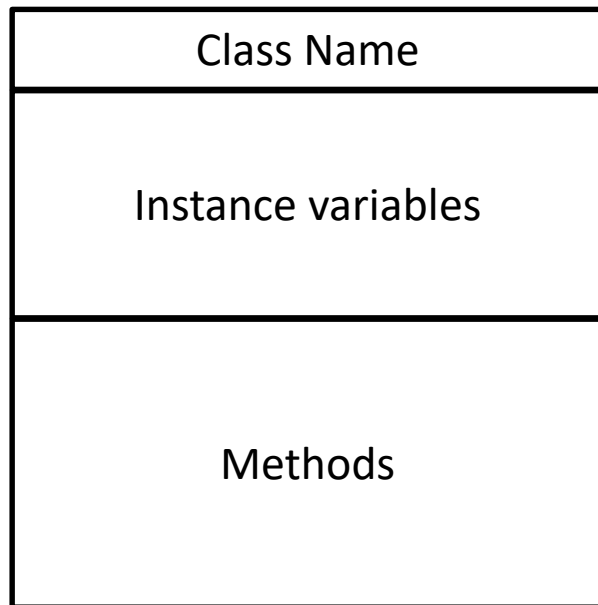
ex) public void reSize(double newSide){...}

→ + resize(double newSide): void

# UML

---

## - Class Diagram



### **Modifier**

- private: minus(-)
- public: plus(+)
- protected: sharp(#)
- package: tilde(~)
- static: underline
- final: ALL CAPITAL LETTERS

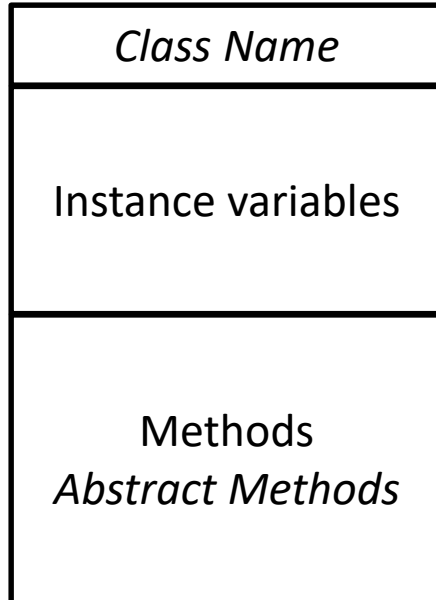


# UML

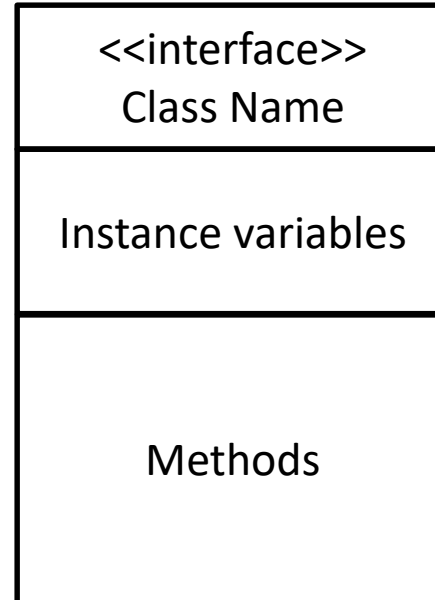
---

## - Class Diagram

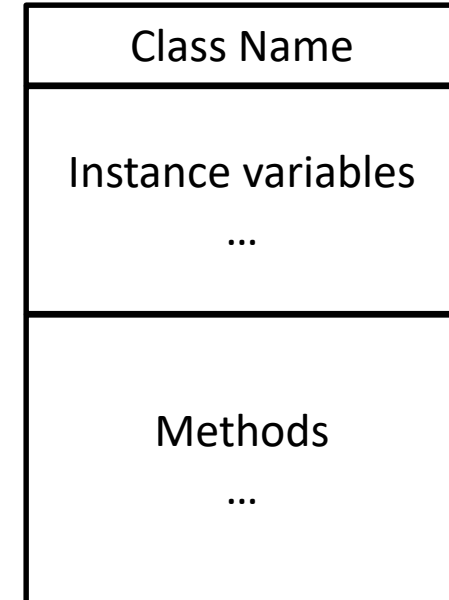
✓ **Abstract:** *italic*



✓ **Interface:** <<interface>>



✓ **Ellipsis:** ...

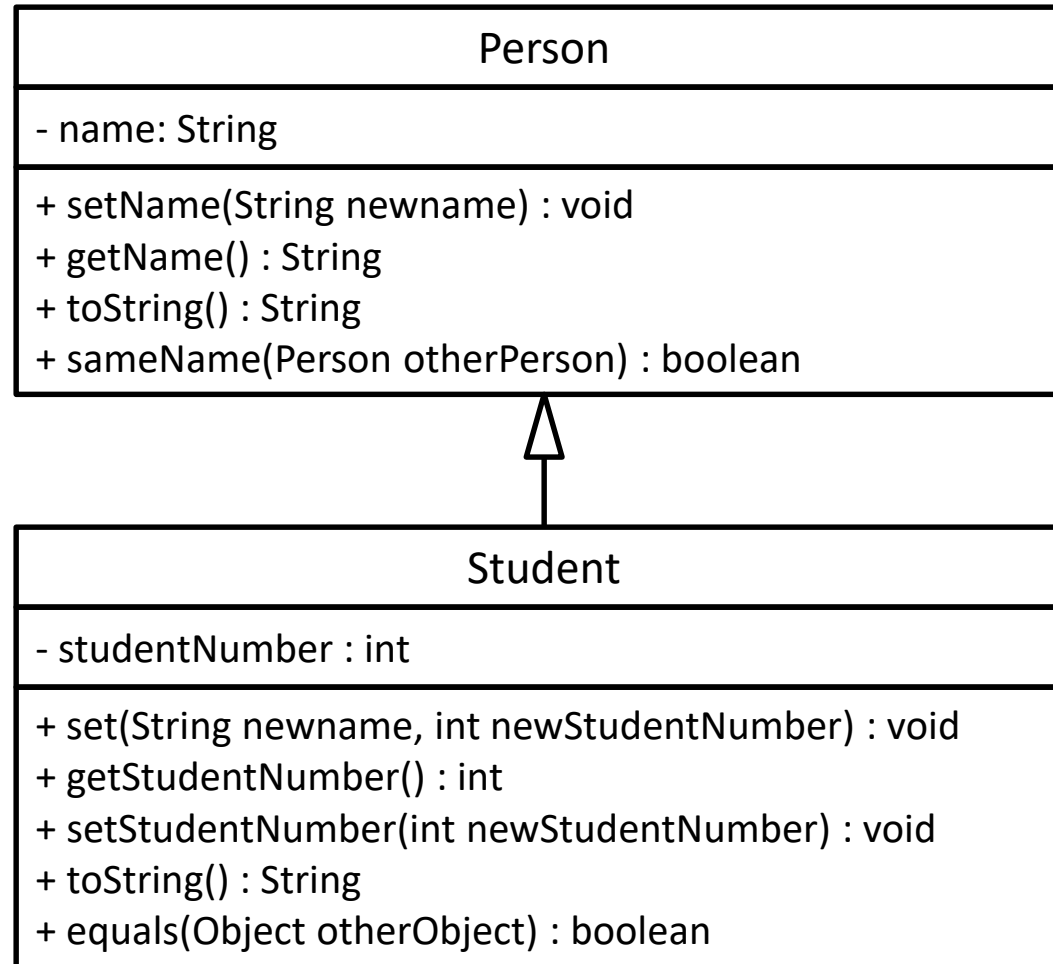


# UML

---

## - Class Diagram

### ✓ Inheritance



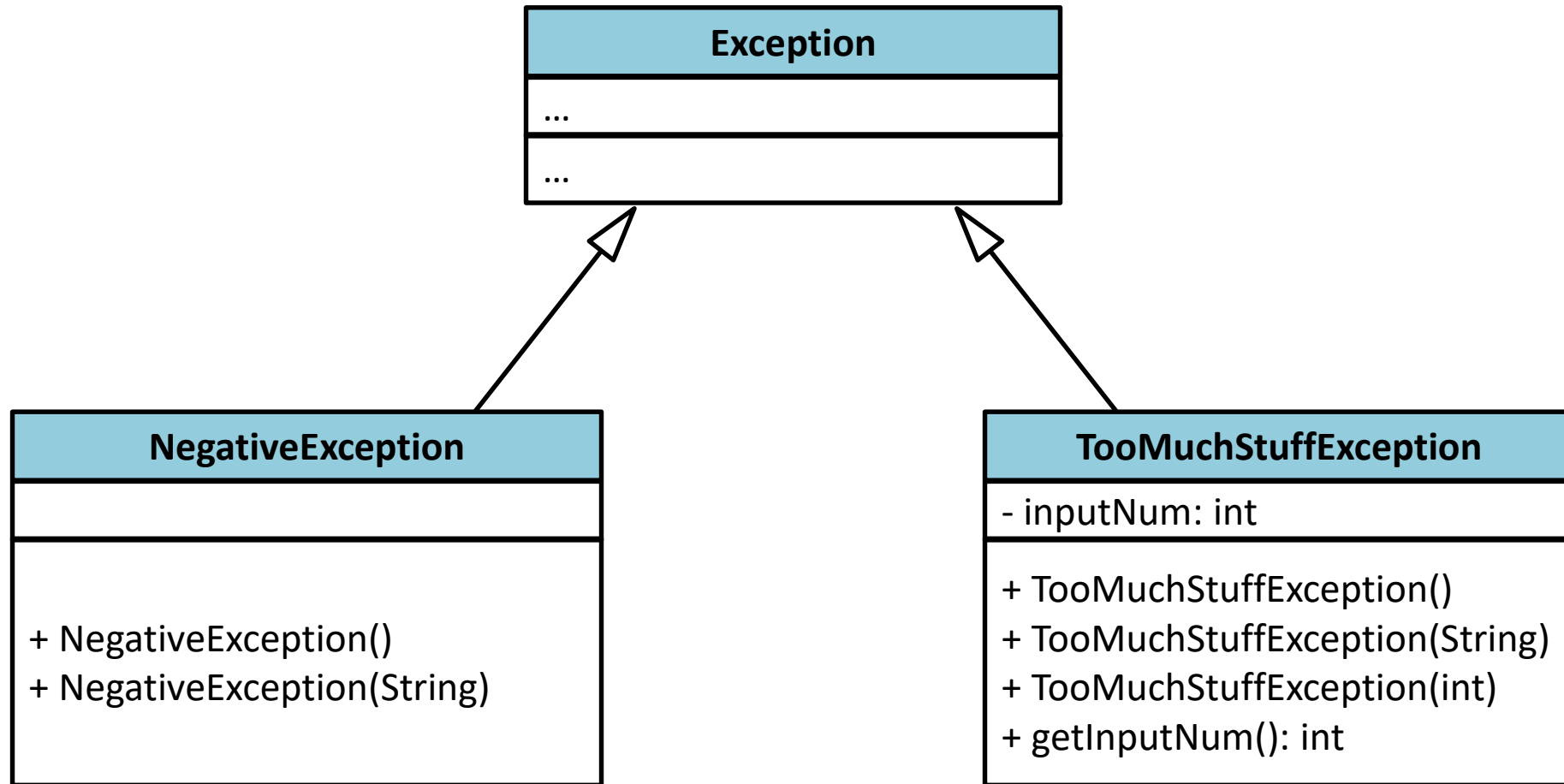
# 과제

---

- NegativeException, TooMuchStuffException, Employee class를 생성한다.
  - NegativeException: 음수 값을 입력하면 발생하는 exception
    - 인자가 없는 생성자 작성
      - default message는 "work time must be positive"
    - String 인자를 받는 생성자 작성 ( 받은 인자를 message로 함 )
  - TooMuchStuffException: 24를 초과하는 숫자를 입력하면 발생하는 exception
    - 한 개의 instance variable을 가짐 ( private int inputNum )
    - 인자가 없는 생성자 작성
      - default message는 "Too much stuff!"
    - String 인자를 받는 생성자 작성 ( 받은 인자를 message로 함 )
    - int 인자를 받는 생성자 작성 ( inputNum을 초기화 )
      - message는 "Too much stuff!"
    - inputNum을 반환하는 getter 작성

# 과제

---



# 과제

## - Employee class

- 3개의 private instance variable를 가진다.
  - ◎ String name
  - ◎ int workDay
  - ◎ int workHours
- name을 인자로 받고 instance variable의 값을 초기화하는 생성자 작성
  - > workDay = 1, workHours = 0으로 초기화
- 3개의 instance variable에 대한 getter를 작성한다.
- workDay를 1만큼 증가시키는 void addWorkDay() method 작성
- workHours를 hours 만큼 증가시키는 void addWorkHours(int hours) method 작성

Employee
- name: String - workDay: int - workHours: int
+ Employee(String) + getName(): String + getWorkDay(): int + getWorkHours(): int + addWorkDay(): void + addWorkHours(int): void

# 과제

---

- ExceptionDemo class를 생성하고 main method에서 다음과 같은 코드를 작성한다.
  - Employee 객체를 하나 생성한다.
  - 무한 반복문을 사용하여 다음 역할을 수행한다.
    1. 해당 employee의 workDay에서의 근무 시간을 입력 받는다. ex) "1일차 근무 시간을 입력하세요 : "
    2. 입력한 시간에 따라 다른 결과를 출력한다.
      1. 0보다 작을 경우, NegativeException() 을 발생시킨다.
      2. 0일 경우, Exception("Program Exit")을 발생시킨다.
      3. 24보다 클 경우, TooMuchStuffException(hours)을 발생시킨다. (hours는 입력 받은 근무시간 값)
      4. 1~24일 경우, 다음 작업을 수행한다.
        1. 해당 직원의 누적 근무 시간(workHours)을 입력한 값만큼 증가시킨다.
        2. 해당 직원의 근무 일차(workDay)를 1 증가시킨다.
        3. 해당 직원의 이름을 출력한다. ex) "이름 : Lee"
        4. 해당 직원의 누적 근무 시간을 출력한다. ex) "누적 근무 시간 : 22"
        5. Exception이 발생하지 않았다는 안내문을 출력한다. ex) "No Exception has been occurred"

# 과제

---

- ExceptionDemo class를 생성하고 main method에서 다음과 같은 코드를 작성한다. (계속)
  - 3. 발생한 예외에 따른 catch문을 작성한다.
    - 1. NegativeException일 경우, 해당 Exception의 Message를 출력한다.
    - 2. TooMuchStuffException일 경우, inputNum과 함께 message를 출력한다.  
ex) "44, Too Much Stuff!"
    - 3. Exception일 경우, 해당 Exception의 Message를 출력하고 프로그램을 즉각 종료한다. ( System.exit(0) 사용 )
  - 4. finally문에서 위의 모든 결과와는 무관하게 항상 출력되는 문자열을 출력한다.  
ex) "End of try-catch statement"

# 과제

---

## < 출력 예시 >

```
1일차 근무 시간을 입력하세요 : -3
work time must be positive
End of try-catch statement
1일차 근무 시간을 입력하세요 : 12
이름: Lee
누적 근무 시간: 12
No Exception has been occurred
End of try-catch statement
2일차 근무 시간을 입력하세요 : 33
33, Too much stuff!
End of try-catch statement
2일차 근무 시간을 입력하세요 : 22
이름: Lee
누적 근무 시간: 34
No Exception has been occurred
End of try-catch statement
3일차 근무 시간을 입력하세요 : 0
Program Exit
```



# 과제 제출

---

- NegativeException, TooMuchStuffException, Employee, ExceptionDemo class를 제출
- Blackboard에 있는 9주차 과제 제출에 제출
- 매주 실습 수업에 있는 과제는 Blackboard에 있는 과제 제출에 제출
- Assignment#1, #2, #3의 코드는 gitlab을 통해서 제출, document는 Blackboard 제출

9주차 과제 문의: [lhs9394@naver.com](mailto:lhs9394@naver.com) (조교 이효식)