

## 8장. 기본클래스, 예외처리



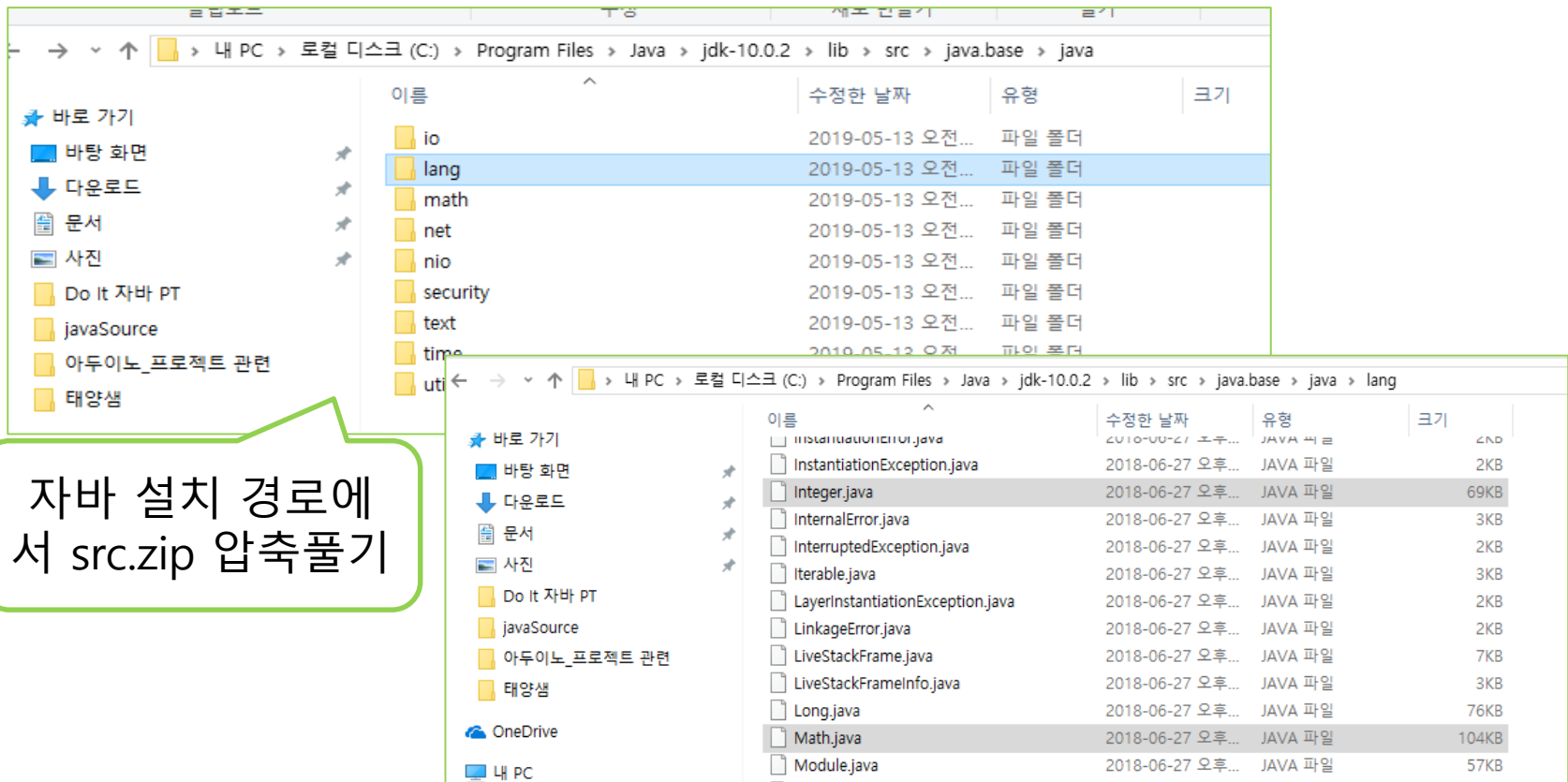
*Object, exceptions*



# java lang 패키지

## ■ java.lang 패키지

- Java.lang.Object / java.lang.String은 어디에 위치할까?
- 포함된 클래스와 인터페이스는 import없이 사용할 수 있다.



자바 설치 경로에서 src.zip 압축풀기

이름	수정된 날짜	유형	크기
io	2019-05-13 오전...	파일 폴더	
lang	2019-05-13 오전...	파일 폴더	
math	2019-05-13 오전...	파일 폴더	
net	2019-05-13 오전...	파일 폴더	
nio	2019-05-13 오전...	파일 폴더	
security	2019-05-13 오전...	파일 폴더	
text	2019-05-13 오전...	파일 폴더	
time	2019-05-13 오전...	파일 폴더	
util	2019-05-13 오전...	파일 폴더	

이름	수정된 날짜	유형	크기
InstantiationException.java	2018-06-27 오후...	JAVA 파일	2KB
Integer.java	2018-06-27 오후...	JAVA 파일	69KB
InternalError.java	2018-06-27 오후...	JAVA 파일	3KB
InterruptedException.java	2018-06-27 오후...	JAVA 파일	2KB
Iterable.java	2018-06-27 오후...	JAVA 파일	3KB
LayerInstantiationException.java	2018-06-27 오후...	JAVA 파일	2KB
LinkageError.java	2018-06-27 오후...	JAVA 파일	2KB
LiveStackFrame.java	2018-06-27 오후...	JAVA 파일	7KB
LiveStackFrameInfo.java	2018-06-27 오후...	JAVA 파일	3KB
Long.java	2018-06-27 오후...	JAVA 파일	76KB
Math.java	2018-06-27 오후...	JAVA 파일	104KB
Module.java	2018-06-27 오후...	JAVA 파일	57KB



# java lang 패키지

## ■ 주요 클래스

클래스	용도
Object	- 자바 클래스의 최상위 클래스로 사용
System	- 표준 입력 장치(키보드)로부터 데이터를 입력받을때 사용 - 표준 출력 장치(모니터)로 출력하기 위해 사용 - 자바 가상 기계를 종료시킬때 사용
Class	- 클래스를 메모리로 로딩할 때 사용
String	- 문자열을 저장하고 여러 가지 정보를 얻을 때 사용
Math	- 수학함수를 이용할 때 사용
<b>Wrapper :</b> Byte, Short, Character, Integer, Float, Double	- 기본 타입의 데이터를 갖는 객체를 만들 때 사용 - 문자열 기본타입으로 변환할때 사용



# Object 클래스

## ▪ 모든 클래스의 최상위 클래스 Object

- Java.lang.Object
- 모든 클래스는 Object 클래스로부터 상속을 받는다. (컴파일러가 자동 변환)

```
class Book {  
    int bookNumber;  
    String bookTitle;  
}
```

코드 작성할때

```
class Book extends Object {  
    int bookNumber;  
    String bookTitle;  
}
```

컴파일러가 변환

생략되어 있음



# Object 클래스

## ▪ Object 클래스의 주요 메서드

메서드	용 도
String <b>toString()</b>	객체를 문자열로 표현하여 반환한다. 재정의하여 객체에 대한 설명이나 특정 멤버 변수 값을 반환한다.
boolean <b>equals(Object obj)</b>	두 인스턴스가 동일한지 여부를 반환한다. 재정의하여 <b>논리적으로 동일한 인스턴스</b> 임을 정의 할 수 있다.
int <b>hashCode()</b>	객체의 해시 코드 값을 반환한다.
Object <b>clone()</b>	객체를 복제하여 동일한 멤버 변수 값을 가진 새로운 인스턴스를 생성한다.



# Object 클래스

## ▪ Object 클래스의 주요 메서드

<b>Module</b> java.base			
<b>Package</b> java.lang			
<b>Class</b> Object			
java.lang.Object			
public class Object			
Class Object is the root of the class hierarchy.			
All Methods	Instance Methods	Concrete Methods	Deprecated Methods
Modifier and Type	Method	Description	
protected Object	clone()	Creates and returns a copy of this object.	
boolean	equals(Object obj)	Indicates whether some other object is "equal to" this one.	
protected void	finalize()	<b>Deprecated.</b> The finalization mechanism is inherently flawed because it does not allow the programmer to know when the finalizer will be invoked, and it is possible that the finalizer will never be invoked at all or that the finalizer will be invoked multiple times.	
Class<?>	getClass()	Returns the runtime class of this Object instance.	
int	hashCode()	Returns a hash code value for the object.	
void	notify()	<b>toString</b>  public String toString()  Returns a string representation of the object. In general, the toString method returns a string that is easy for a person to read. It is recommended that all subclasses override this method.  The toString method for class Object returns a string consisting of the class's name followed by the hash code of the object. In other words, this method returns a string in the form: <code>toString()</code>	
void	notifyAll()		
String	toString()		



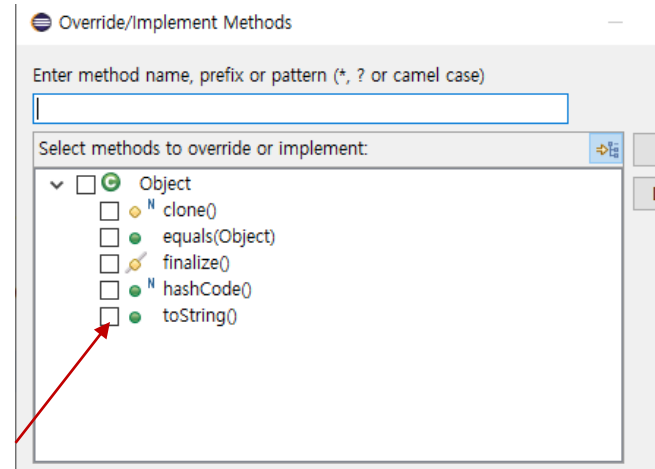
# toString() 메서드

## ■ toString() 메서드

- 객체 정보를 문자열로 바꾸어 주는 기능을 함
- Integer나 String 등 클래스는 toString() 메서드가 이미 재정의 되어 있음.
- 사용자 정의 클래스는 toString() 재정의 해야함

```
package object;
```

```
public class Book {  
    int bookNumber;  
    String bookTitle;  
  
    Book(int bookNumber, String bookTitle){  
        this.bookNumber = bookNumber;  
        this.bookTitle = bookTitle;  
    }  
  
    @Override  
    public String toString() {  
        return bookNumber + ", " + bookTitle;  
    }  
}
```



우클릭 > source >  
Override/Implement Methods



# toString() 메서드

## ■ toString() 메서드

```
public class ToStringEx {  
  
    public static void main(String[] args) {  
        //String으로 생성한 인스턴스는 문자열로 출력  
        String name = new String("홍길동");  
        System.out.println(name);  
        System.out.println(name.toString());  
  
        //Book으로 생성한 문자열은 toString() 재정의 해야함  
        Book book = new Book(100, "개미");  
        System.out.println(book);  
        System.out.println(book.toString());  
    }  
}
```

```
public String toString() {  
    return this;  
}
```

이미 재정의 되어있음

홍길동  
홍길동  
개미,100  
개미,100





# equals() 메서드

## ▪ equals() 메서드

- 두 인스턴스의 주소 값을 비교하여 boolean 값(true/false)를 반환
- 재정의하여 두 인스턴스가 논리적으로 동일함의 여부를 반환.
- 같은 번호의 책인 경우 여러 인스턴스의 주소값은 다르지만, 같은 책으로 인정할 수 있으므로 equals() 메서드를 재정의해야 함.
- **instanceof** 키워드 사용

```
@Override
public boolean equals(Object obj) {
    if(obj instanceof Book) {
        Book book = (Book)obj; //강제 타입 변환
        if(this.bookNumber == book.bookNumber)
            return true;
    }
    return false;
}
```



# equals() 메서드

## ▪ equals() 메서드

```
public class EqualsTest {  
    public static void main(String[] args) {  
        //String으로 생성한 인스턴스의 메모리 주소와 값 비교  
        String name1 = new String("홍길동");  
        String name2 = new String("홍길동");  
        System.out.println(name1==name2); //메모리 주소는 다르다.  
        System.out.println(name1.equals(name2)); //저장된 값은 같다.  
  
        //Book으로 생성한 인스턴스의 메모리 주소와 값 비교  
        Book book1 = new Book(100, "개미");  
        Book book2 = new Book(100, "개미");  
        System.out.println(book1==book2);  
        System.out.println(book1.equals(book2));  
    }  
}
```

false  
true  
=====  
false  
false

Book 클래스에서  
는 equals() 메서드  
재정의 필요



# equals() 메서드

## ▪ equals() 재정의(Override)

```
public class Book {  
    int bookNumber;  
    String bookTitle;  
  
    Book(int bookNumber, String bookTitle){  
        this.bookNumber = bookNumber;  
        this.bookTitle = bookTitle;  
    }  
  
    @Override  
    public boolean equals(Object obj) {  
        if(obj instanceof Book) {  
            Book book = (Book)obj; //강제 타입 변환  
            if(this.bookNumber == book.bookNumber)  
                return true;  
        }  
        return false;  
    }  
}
```



# 실습 문제 1 – Object 클래스

다음 코드의 출력 결과가 "진돗개 백구"가 되도록 MyDog 클래스를 수정하세요.

```
class MyDog{
    String name;
    String type;

    public MyDog(String name, String type) {
        this.name = name;
        this.type = type;
    }

    ...
}

public class ToStringTest {

    public static void main(String[] args) {

        MyDog dog = new MyDog("백구", "진돗개");
        System.out.println(dog);
    }

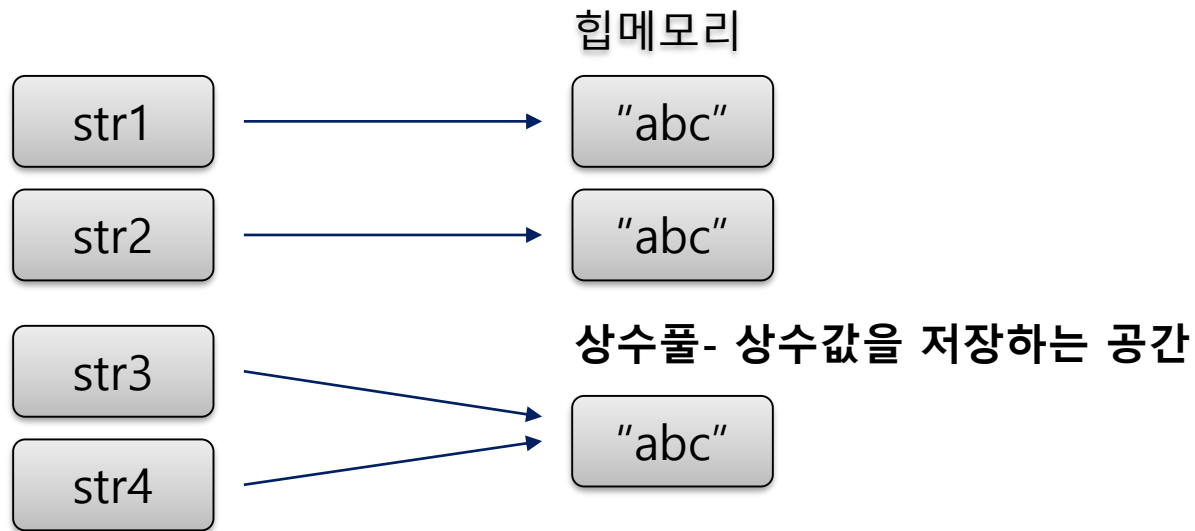
}
```



# String 클래스

## ● String 클래스

- 문자열을 저장하고 여러 가지 정보를 얻을 때 사용하는 클래스
- 문자열을 사용하여 생성자의 매개변수로 하여 생성하는 방식과 이미 생성된 문자열 상수를 가리키는 방식이 있다.



# String 클래스

## ● String 객체 생성

```
public class StringTest {  
    public static void main(String[] args) {  
  
        String str1 = new String("abc");  
        String str2 = new String("abc");  
  
        //인스턴스가 매번 새로 생성되므로 주소값이 다름  
        System.out.println(str1==str2);  
        //문자열 값은 같음  
        System.out.println(str1.equals(str2));  
  
        String str3 = "abc";  
        String str4 = "abc";  
  
        //문자열 "abc"는 상수 풀에 저장되어 있어서 주소 값이 같음  
        System.out.println(str3==str4);  
        System.out.println(str3.equals(str4));  
    }  
}
```

```
false  
true  
true  
true
```



# String 클래스

## ● String의 주요 메서드

메서드	기능 설명
charAt(index)	- 매개값으로 전달된 index의 문자를 반환함
indexOf(문자열)	- 매개값으로 주어진 문자열이 시작되는 인덱스를 리턴
lastIndexOf(문자열)	- 매개값으로 주어진 문자열을 뒤에서 검색하여 인덱스를 리턴
substring(begin, end)	- 매개값의 시작 인덱스부터 끝(끝-1)까지 문자 추출
replace(변경전, 변경후)	- 문자열을 변경(수정)
split(구분기호)	- 구분기호로 문자열을 분리하여 리스트로 반환
toUpperCase()	- 영어 문자열을 대문자로 변환
toLowerCase()	- 영어 문자열을 소문자로 변환



# String의 주요 메서드

## ● charAt() – 문자추출

매개값으로 주어진 인덱스의 문자를 리턴함

```
String msg = "행운을 빌어요!!";

char ch = msg.charAt(0);
System.out.println(ch); //행

//주민등록번호에서 남여를 구분
String jumin = "020815-4234567";
char gender = jumin.charAt(7); //성별

System.out.println(gender); //4

switch(gender) {
case '1': case '3': //주의!! - char형 이므로 문자로
    System.out.println("남자입니다.");
    break;
case '2': case '4':
    System.out.println("여자입니다.");
    break;
default:
    System.out.println("지원되지 않는 기능입니다.");
    break;
}
```





# String의 주요 메서드

## ● indexOf() – 문자열 찾기

매개값으로 주어진 문자열이 시작되는 인덱스를 리턴합니다.

찾지 못하면 -1을 리턴함

```
String subject = "자바 프로그래밍 입문";
int location1 = subject.indexOf("프로그래밍");
System.out.println(location1);

int location2 = subject.indexOf("코딩");
System.out.println(location2); //-1

//문자열 검색
if(subject.indexOf("자바") != -1) {
    System.out.println("자바와 관련된 책이군요!!");
}else {
    System.out.println("자바와 관련이 없는 책이군요!!");
}
```



# String의 주요 메서드

- **lastIndexOf() – 문자열 찾기**

매개값으로 주어진 문자열을 뒤에서 검색하여 인덱스를 리턴합니다.  
찾지 못하면 -1을 리턴함

```
String url = "http://www.korea.kr/boards";  
int n = url.lastIndexOf("/");  
  
System.out.println(n); //19  
System.out.println(url.substring(n));
```



# String의 주요 메서드

## ● substring() – 문자열 잘라내기

1. substring(beginIdx, endIdx)
  - 매개값의 시작 인덱스부터 끝(끝-1)까지 문자 추출
2. substring(beginIdx)
  - 시작인덱스부터 문자열의 끝까지 문자 추출

```
String juminNum = "991125-1234567";  
  
String firstNum = juminNum.substring(0, 6);  
System.out.println(firstNum);  
  
String secondNum = juminNum.substring(7);  
System.out.println(secondNum);
```



# String의 주요 메서드

- **replace()** – 문자열 수정(대체)

```
//replace(변경전 문자, 변경후 문자)
// 문자 대체
String str1 = "Hello Java!";
String result1 = str1.replace('a', '@');
System.out.println(result1); // Hello J@v@!

// 문자열 대체
String str2 = "한국의 수도는 서울이다.";
String result2 = str2.replace("한국", "대한민국");
System.out.println(result2); // 대한민국의 수도는 서울이다.

// 특수 문자 제거
String phone = "010-1234-5678";
String result3 = phone.replace("-", "");
System.out.println(result3); // 01012345678
```



# String의 주요 메서드

## ● split() – 문자열 분리

```
//split(구분기호) - 구분기호로 구분하여 배열로 반환함
String carts = "potato strawberry garlic";
String[] array = carts.split(" ");

System.out.println(array[0]); //potato
System.out.println(array[1]); //strawberry

for(int i = 0; i < array.length; i++){
    System.out.print(array[i]+ " ");
}
/*
    실습) 문자열을 분리한 후 출력하기
        123
        456
        789
*/
String str = "1,2,3,4,5,6,7,8,9";
String[] array2 = str.split(",");

System.out.println(array2[0]); //1

//코드 작성
```



# String의 주요 메서드

- toUpperCase(), toLowerCase()

```
/*
 * toUpperCase() - 대문자로 변환
 * toLowerCase() - 소문자로 변환
 * equals(str2) - 대소문자 구분하여 일치 여부
 * equalsIgnoreCase(str2) - 대소문자 구분없이 일치 여부
 */
String s1 = "Hello World!";
String s2 = "HELLO WORLD!";

if(s1.equals(s2))
    System.out.print(s1.toUpperCase());
else if(s1.equalsIgnoreCase(s2))
    System.out.println(s1.toLowerCase()); //hello world!
else
    System.out.println(s2);
```



# 정규 표현식(Regular Expression)

## ❖ 정규 표현식(Regular Expression)

- 특정한 규칙을 가진 문자열의 집합을 표현하고 처리하는 것을 말한다.
- 데이터의 유효성 검사에 주로 사용된다.
- 자주 사용하는 정규 표현식

표현식	설 명
<code>^[0-9]*\$</code>	숫자
<code>^[a-zA-Z]*\$</code>	영문 대, 소문자
<code>^[가-힣]*\$</code>	한글
<code>^010[-](d{3} d{4})[-]d{4}\$</code>	휴대폰
<code>^d{6}[-][1-4]{6}\$</code>	주민등록번호
<code>^d{3}[-]d{2}\$</code>	우편번호



# 정규 표현식(Regular Expression)

## ❖ 정규 표현식(Regular Expression)

### ▪ 자주 사용하는 정규 표현식

메타문자	설 명
^	정규식 시작
\$	정규식 끝
*	문자 반복(+ 도 가능)
{3}	반복횟수
[x]	x를 찾음
\d	숫자
\s	공백
\w	알파벳+숫자





# 정규 표현식(Regular Expression)

## ❖ 정규 표현식(Regular Expression)

메타 문자	설명	예시 ( <code>Pattern.compile()</code> 인자)
<code>*</code>	0번 이상 반복	<code>"a*b"</code> → <code>"b"</code> , <code>"ab"</code> , <code>"aaab"</code>
<code>+</code>	1번 이상 반복	<code>"a+b"</code> → <code>"ab"</code> , <code>"aaab"</code> (but not <code>"b"</code> )
<code>?</code>	0번 또는 1번	<code>"a?b"</code> → <code>"b"</code> , <code>"ab"</code>
<code>^</code>	문자열의 시작	<code>"^a"</code> → <code>"a"</code> 로 시작하는 문자열
<code>\$</code>	문자열의 끝	<code>"a\$"</code> → <code>"a"</code> 로 끝나는 문자열
<code>[0-9]</code>	0부터 9까지의 숫자	<code>"[0-9]+"</code> → <code>"123"</code>
<code>[a-z]</code>	소문자 a부터 z까지	<code>"[a-z]+"</code> → <code>"hello"</code>
<code>\d</code>	숫자 ( <code>[0-9]</code> 와 동일)	<code>"\d+"</code> → <code>"123"</code>
<code>\w</code>	단어 문자 (알파벳, 숫자, <code>_</code> )	<code>"\w+"</code> → <code>"abc_123"</code>



# 정규 표현식(Regular Expression)

## ❖ 정규 표현식(Regular Expression)

클래스	메서드	기능 설명
Pattern	compile(regex)	- 정규 표현식을 컴파일(해석) 한다.
	matches(regex, str)	- 문자열이 정규표현식에 매칭(일치) 여부를 반환함(true/false)
Matcher	matches(문자열)	- Pattern 클래스로 compile한 정규표현식 객체로 문자열 객체 생성
	matches()	- Matcher 클래스의 객체가 정규표현식에 일치하는 여부를 반환함(true/false)
String	replaceAll(regex, 대체문자)	- 특정 패턴과 일치하는 모든 부분을 변경



# 정규 표현식(Regular Expression)

## ❖ Pattern, Matcher 클래스 활용

```
//a*b 패턴 검사 - a가 0번 이상 반복
//a+b 패턴 - a가 1번 이상 반복
Pattern pat = Pattern.compile("a*b");
Matcher m = pat.matcher("aaab"); //a가 없어도 true
boolean b1 = m.matches();

System.out.println(b1);

//숫자만 허용하는 패턴 검사
String pattern = "[0-9]*$";
String str = "abc1031";

boolean b2 = Pattern.matches(pattern, str);
System.out.println(b2);
```

```
true
false
```



# 정규 표현식(Regular Expression)

## ❖ Pattern, Matcher 클래스 활용

```
//한글 이름과 전화번호 패턴 검사
String name = "을지문덕장군";
String tel = "010-1234-5678";

boolean name_check = Pattern.matches("^[가-힝]{2,5}$", name); //한글 2~5자 제한
boolean tel_check = Pattern.matches("^010[-](\\d{3}|\\d{4})[-]\\d{4}$", tel);

System.out.println("이름 검사: " + name_check);
System.out.println("전화번호 검사: " + tel_check);

//한글 이름 패턴 유효성 검사
Scanner sc = new Scanner(System.in);
System.out.print("한글 이름을 입력하세요: ");
String inputName = sc.nextLine();

if (!Pattern.matches("^[가-힝]{2,5}$", inputName)) {
    System.out.println("올바른 한글 이름이 아닙니다!");
}
System.out.println("이름: " + inputName);
```

```
true
false
이름 검사: false
전화번호 검사: true
한글 이름을 입력하세요: 을지문덕장군
올바른 한글 이름이 아닙니다!
이름: 을지문덕장군
```



# 정규 표현식(Regular Expression)

## ❖ replaceAll() 메서드 활용

```
//비밀번호 보안 처리
String password = "P@ssw0rd!";

//replaceAll(정규표현식, 대체할 문자)
//^ - 부정 문자(아니다)
String masked = password.replaceAll("[^a-zA-Z0-9]", "*");

System.out.println(masked); // P*ssw0rd*

//게시글 금칙어 처리
String text = "안녕@하세요! #스팸";
// 한글과 공백만 허용
String filtered = text.replaceAll("[^ㄱ-힣\\s]", "*");

System.out.println(filtered); // 안녕*하세요* **

//실습 예제
String str = "THANK!@YOU%/";
String res = str.replaceAll("[^ㄱ-하- |가-힣a-zA-Z0-9]", "*");
System.out.println(res);
```



## 실습 문제 2 – 정규 표현식

아래 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
String str = "THANK!@YOU%/";  
String res = str.replaceAll("[^ㄱ-하- | 가-힣a-zA-Z0-9]", "*");  
System.out.println(res);
```



# Wrapper 클래스

## ● 기본 자료형을 위한 클래스

- 기본 자료형을 멤버 변수로 포함하여 메서드를 제공함으로써 기본자료형의 객체를 제공하는 클래스.
- 기본 자료형을 감쌌다는 의미로 Wrapper 클래스라고 한다.

지금까지 정수를 사용할 때 기본형인 int를 사용했다. 그런데 정수를 객체형으로 사용해야 하는 경우가 발생한다.

```
public void setValue(Integer i){.....}  
//객체를 매개변수로 받는 경우
```

```
public Integer returnValue(){.....}  
//반환값이 객체인 경우
```

기본형	Wrapper 클래스
boolean	Boolean
byte	Byte
char	Character
int	Integer
double	Double



# Wrapper 클래스

## ■ 자료형의 크기

```
public class TypeSize {  
    public static void main(String[] args) {  
        //자료형의 크기 확인  
        System.out.printf("byte형의 크기      ==> %d\n", Byte.SIZE);  
        System.out.printf("short형의 크기     ==> %d\n", Short.SIZE);  
        System.out.printf("int형의 크기      ==> %d\n", Integer.SIZE);  
        System.out.printf("long형의 크기     ==> %d\n", Long.SIZE);  
        System.out.printf("float형의 크기    ==> %d\n", Float.SIZE);  
        System.out.printf("double형의 크기   ==> %d\n", Double.SIZE);  
        System.out.printf("char형의 크기     ==> %d\n", Character.SIZE);  
    }  
}
```

byte형의 크기	==> 8
short형의 크기	==> 16
int형의 크기	==> 32
long형의 크기	==> 64
float형의 크기	==> 32
double형의 크기	==> 64
char형의 크기	==> 16





# Wrapper 클래스

## Integer 클래스

```
public class IntegerTest {  
  
    public static void main(String[] args) {  
        //int형 < Integer형 (자동 형변환)  
        Integer num1 = 100;  
        int num2 = 200;  
        int sum;  
  
        //intValue() -> Integer형을 int형으로 변환함  
        sum = num1.intValue() + num2;  
        System.out.println(sum);  
  
        //valueOf() -> 정수나 문자열을 Integer 클래스로 변환함  
        Integer n1 = Integer.valueOf("100");  
        System.out.println(n1);  
  
        //parseInt() -> 매개로 전달된 문자열을 정수형으로 변환  
        int n2 = Integer.parseInt("200");  
        System.out.println(n2);  
  
        //parseDouble() -> 문자열을 실수형으로 변환함  
        double dNum = Double.parseDouble("1.609");  
        System.out.println(dNum);  
    }  
}
```

```
300  
100  
200  
1.609
```



# 난수 생성

## ● 난수 생성 비교

방법	특징	사용 예시
Math.random()	$0.0 \leq x < 1.0$ (Double)	<code>(int)(Math.random() * 6) + 1</code>
Random 클래스	다양한 자료형 지원	<code>rnd.nextInt(6) + 1</code>

## ● Random 클래스

난수(무작위수)를 생성하는 클래스이다.

### Random

```
public Random(long seed)
```

Creates a new random number generator using a single

**Implementation Requirements:**

The invocation `new Random(seed)` is equivalent to:

```
Random rnd = new Random();  
rnd.setSeed(seed);
```

**Parameters:**

**seed** - the initial seed

**See Also:**

`setSeed(long)`



# Random 클래스

## ● Random 클래스 활용

```
//난수 생성 - Math.random()
int n1 = (int)(Math.random() * 2); //0 ~ 1
System.out.println(n1);

//난수 생성 - Random() 클래스 활용
Random rnd = new Random();
//rnd.setSeed(0); //시드 설정 - 동일한 결과 반복

//시스템 시간을 기반으로 자동 변화
System.out.println(rnd.nextInt());
int n2 = rnd.nextInt(2);
System.out.println(n2); //0 ~ 1

//동전 던지기
int coin = rnd.nextInt(2) + 1;
if(coin == 1)
    System.out.println("앞면");
else
    System.out.println("뒷면");
```



# Random 클래스

## ● Random 클래스를 활용한 주사위 던지기 게임

```
Random random = new Random();
int dice = random.nextInt(6) + 1;
//System.out.println(dice);

int dice1, dice2, total;
for(int i=0; i<10; i++) {
    dice1 = random.nextInt(6) + 1;
    dice2 = random.nextInt(6) + 1;
    total = dice1 + dice2;
    System.out.println(total);
    if(total==7)
        System.out.println("Seven Thrown!!");
    if(total==10)
        System.out.println("Ten Thrown!!");
    if(dice1==dice2)
        System.out.println("Double Thrown!!");
}
```

```
7
Seven Thrown!!
4
Double Thrown!!
7
Seven Thrown!!
6
4
Double Thrown!!
7
Seven Thrown!!
6
6
8
4
```



# 숫자 추측 게임

## ● 숫자 추측 게임

숫자(1~30)를 입력하세요:20  
너무 작아요!  
숫자(1~30)를 입력하세요:25  
너무 작아요!  
숫자(1~30)를 입력하세요:27  
정답!

```
Scanner sc = new Scanner(System.in);
Random rand = new Random();
int comNum = rand.nextInt(30) + 1; //1 ~ 30

//System.out.println(com);
while(true) {
    System.out.print("숫자(1~30)를 입력하세요:");
    int guessNum = sc.nextInt();
    if(guessNum == comNum) {
        System.out.println("정답!");
        break;
    }else if(guessNum > comNum) {
        System.out.println("너무 커요!");
    }else {
        System.out.println("너무 작아요!");
    }
}
sc.locale();
```



# 문자열 자르기 – split()

- String 클래스의 split() 사용

```
public class SplitTest {  
    public static void main(String[] args) {  
        //문자열 자르기  
        String str = "봄 여름 가을 겨울";  
        String[] season = str.split(" "); //배열 생성  
  
        Random rnd = new Random();  
        int rndIdx = rnd.nextInt(season.length);  
  
        System.out.println(season[rndIdx]);  
    }  
}
```



# 영어 타자 게임

## ● 영어 타자 게임

- 게임이 시작되면 영어 단어가 화면에 표시된다.
- 사용자는 최대한 빠르고 정확하게 입력해야 한다.
- 바르게 입력했으면 다음 문제로 넘어가고 "통과"를 출력한다.
- 오타가 있으면 같은 단어가 한 번 더 나온다.
- 타자 게임 시간을 측정한다.

[타자 게임], 준비되면 엔터!

문제 번호 1  
bird  
bird  
통과!!

문제 번호 2  
mountain  
mountain  
통과!!

문제 번호 3  
earth  
earth  
통과!!

문제 번호 8  
galaxy  
gala  
오타! 다시 도전!

문제 번호 8  
moon  
moon  
통과!!

문제 번호 9  
moon  
moon  
통과!!

문제 번호 10  
mountain  
mountain  
통과!!

게임 소요시간 22.10초



# 영어 타자 게임

## ● 영어 타자 게임

```
import java.util.Random;
import java.util.Scanner;
public class TypingGame {
    public static void main(String[] args) {
        String str = "sun moon earth galaxy mountain flower tree bird girl man";
        int n = 1; //문제 번호
        long start, end; //시작, 종료 시간
        double elapsedTime;

        Scanner sc = new Scanner(System.in);
        System.out.print("[타자 게임], 준비되면 엔터!");
        sc.nextLine();
        start = System.currentTimeMillis(); //게임 시작시간

        while(n <= 10) {
            //문자열 잘라서 배열 생성 후 랜덤 추출
            String[] words = str.split(" ");
            Random rnd = new Random();
            int rndIdx = rnd.nextInt(words.length);
            String question = words[rndIdx];
```





# 영어 타자 게임

## ● 영어 타자 게임

```
System.out.println("\n문제 번호 " + n);
System.out.println(question); //문제 출제

String answer = sc.nextLine(); //사용자 입력
if(question.equals(answer)) {
    System.out.println("통과!!");
    n++; //다음 문제 번호
}else {
    System.out.println("오타! 다시 도전!");
}
}
end = System.currentTimeMillis(); //게임 종료시간

elapsedTime = (double)(end - start) / 1000;
//서식 문자 사용 : %d-정수, %f-실수, %s-문자열
System.out.printf("게임 소요시간 %.2f초", elapsedTime);

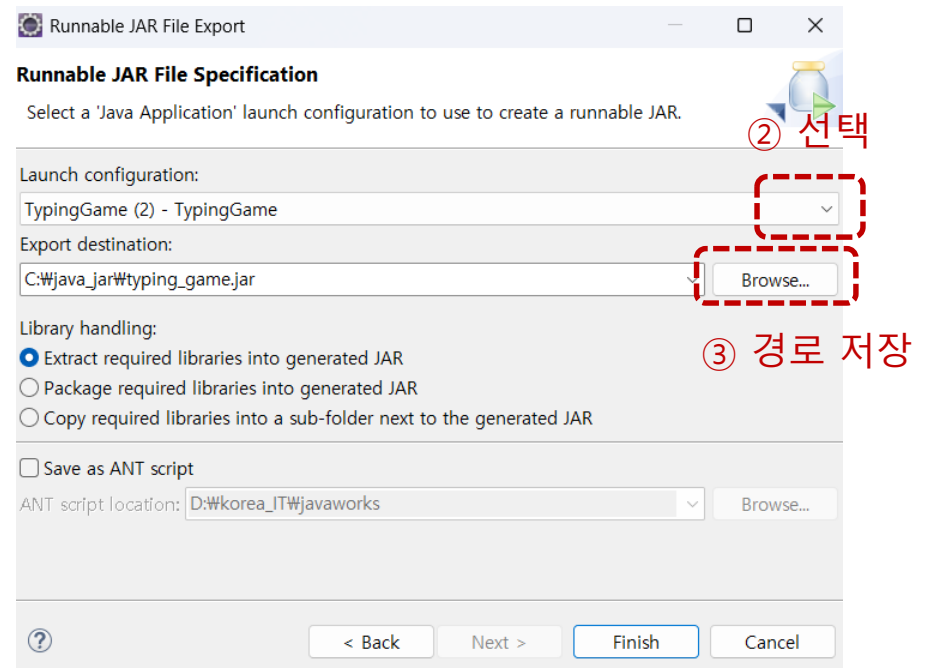
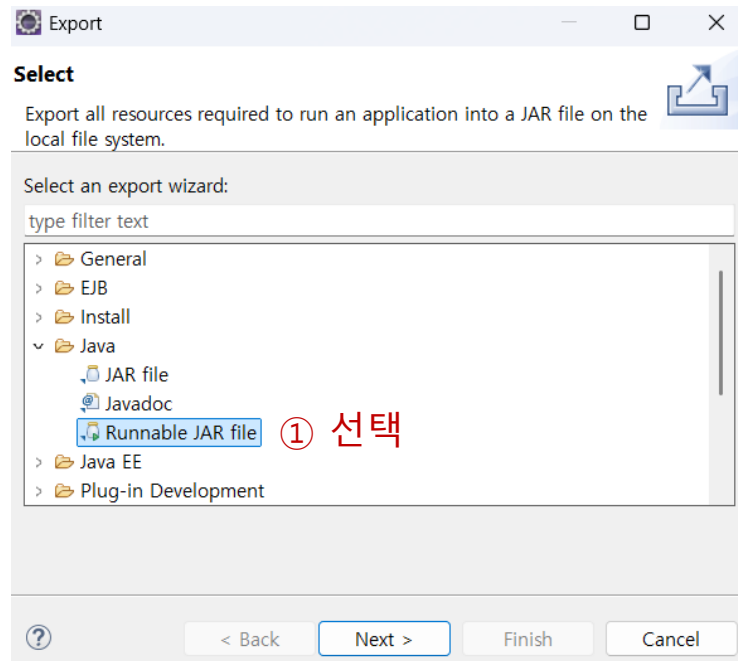
sc.close();
}
}
```



# jar 파일 배포

## ● Jar파일 만들기

프로젝트 이름 - 우측메뉴 - Export - Java - Runnable JAR File



# Jar 파일 명령 프롬프트에서 실행

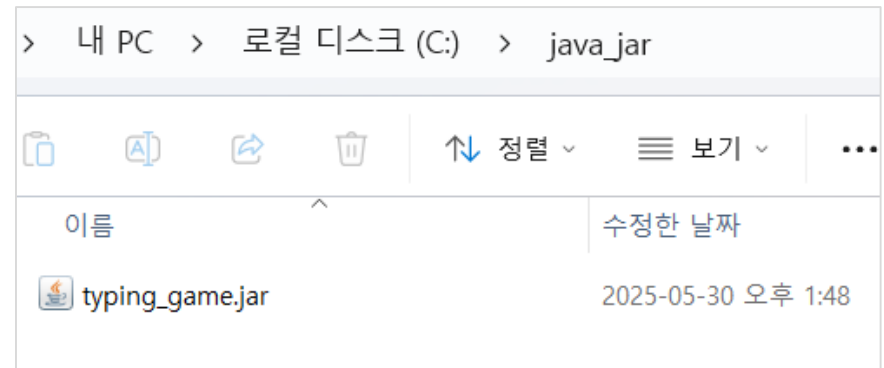
- Jar파일 실행 : `java -jar typing_game.jar`

```
c:\java_jar>java -jar typing_game.jar  
[타자 게임], 준비되면 엔터!
```

```
문제 번호 1  
sun  
sun  
통과!!
```

```
문제 번호 2  
moon  
moon  
통과!!
```

```
문제 번호 3  
tree  
tree  
통과!!
```



# 에러와 예외

## ■ 예외(Exception)

- 사용자의 잘못된 조작 또는 개발자의 잘못된 코딩으로 인한 오류
- 예외가 발생되면 프로그램이 종료되고, 예외 처리를 하면 정상으로 돌아갈 수 있음

## ■ 예외의 종류

### 1. 일반 예외(Exception)

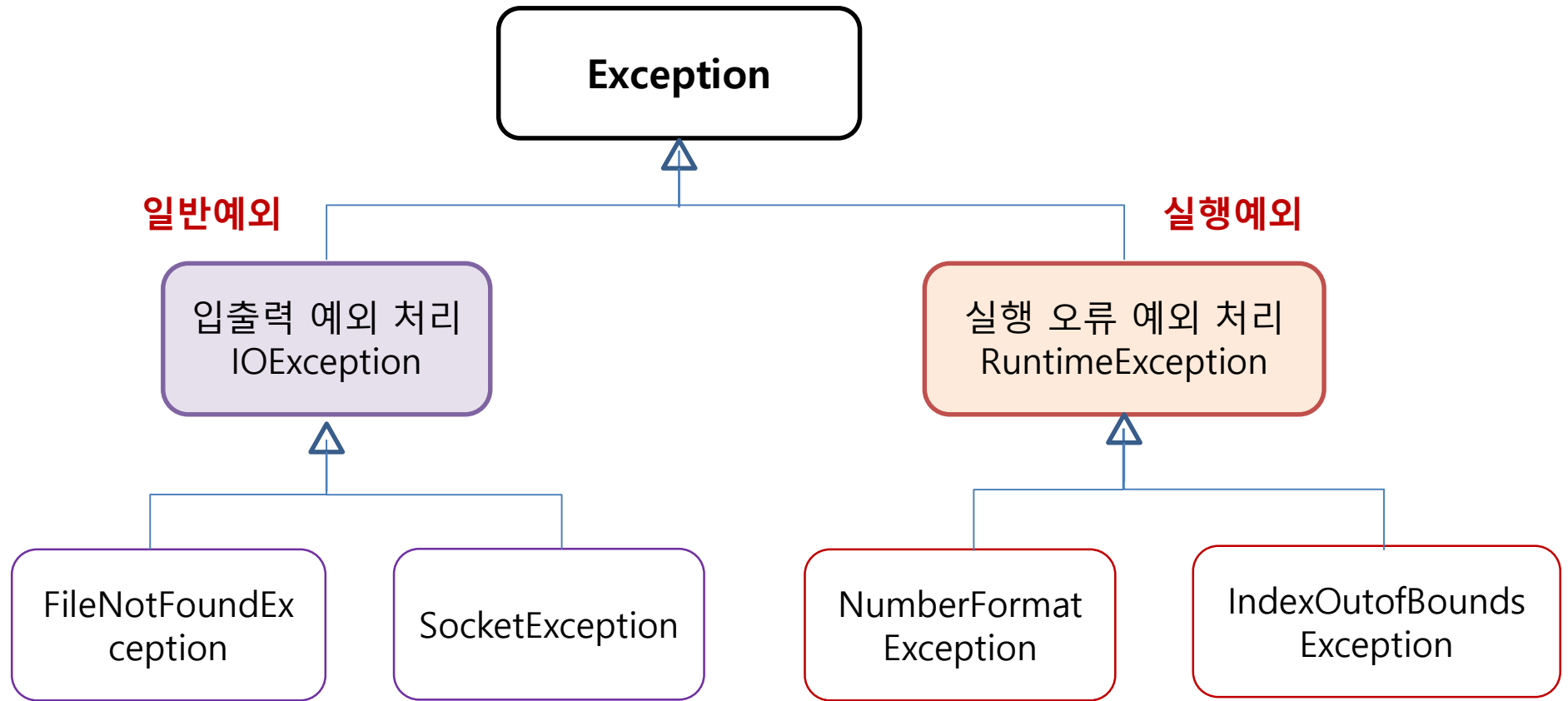
- 예외 처리가 없으면 컴파일 되지 않는 예외 – 컴파일 체크

### 2. 실행 예외(RuntimeException)

- 예외 처리를 생략해도 컴파일 되는 예외
- 개발자의 경험과 판단으로 예외 코드 작성 필요



# 예외 클래스의 종류



# 예외 클래스의 종류

- Java.lang 패키지 -> Exception Summary

Exception Summary	
Exception	Description
<b>ArithmeticException</b>	Thrown when an exceptional arithmetic condi
<b>ArrayIndexOutOfBoundsException</b>	Thrown to indicate that an array has been acc
<b>ArrayStoreException</b>	Thrown to indicate that an attempt has been r
<b>ClassCastException</b>	Thrown to indicate
<b>ClassNotFoundException</b>	Thrown when an a
<b>CloneNotSupportedException</b>	Thrown to indicate
<b>EnumConstantNotPresentException</b>	Thrown when an a
<b>Exception</b>	The class Excepti

**Module** java.base

**Package** java.lang

## Class ArithmeticException

java.lang.Object

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.ArithmeticException

All Implemented Interfaces:

Serializable



# try ~ catch문

- try ~ catch문

예외처리를 하면 예외 상황을 알려 주는 메시지를 볼 수 있고, 프로그램이 비정상적으로 종료되지 않고 계속 수행되도록 만들 수 있다.

```
try{  
    예외가 발생할 수 있는 코드  
}  
catch(처리할 예외 타입 e){  
    예외를 처리하는 코드  
}
```



# 실행 예외

## ● 실행 예외 – 컴파일러가 체크해주지 않음

```
3 public class ExceptionHandling1 {
4
5     public static void printLength(String data) {
6         int result = data.length();
7         System.out.println("문자 수: " + result);
8     }
9
10    public static void main(String[] args) {
11        System.out.println("[프로그램 시작]\n");
12
13        printLength("Hello");
14        printLength(null);
15
16        System.out.println("[프로그램 종료]");
17    }
```

예외가 발생하면  
그 즉시 프로그램  
이 종료됨

[프로그램 시작]

문자 수: 5

Exception in thread "main" [java.lang.NullPointerException](#): Cannot invoke "String  
at exceptions.ExceptionHandling1.printLength([ExceptionHandling1.java:6](#))  
at exceptions.ExceptionHandling1.main([ExceptionHandling1.java:14](#))





# 실행 예외

## ● 실행 예외 처리 – try ~ catch 구문

```
public class ExceptionHandling2 {  
  
    public static void printLength(String data) {  
        //예외 처리: try ~ catch 구문  
        try {  
            int result = data.length();  
            System.out.println("문자 수: " + result);  
        } catch (NullPointerException e) {  
            System.out.println(e.getMessage());  
            e.printStackTrace(); //예외를 추적하면서 출력함(많이 사용됨)  
        }  
    }  
}
```

예외가 발생하면 예외 처리 메시지가 출력되고 프로그램이 실행되고 종료됨

[프로그램 시작]

문자 수: 5

Cannot invoke "String.length()" because "data" is null

java.lang.NullPointerException: Cannot invoke "String.length()" because "data"  
at exceptions.ExceptionHandling2.printLength(ExceptionHandling2.java:8)  
at exceptions.ExceptionHandling2.main(ExceptionHandling2.java:20)

[프로그램 종료]



# 일반 예외

- 일반 예외 – 컴파일러가 체크 함

```
public class ExceptionHandling3 {  
    public static void main(String[] args) {  
        Class.forName("java.lang.String");  
    }  
}
```

Unhandled exception type ClassNotFoundException  
2 quick fixes available:  
• Add throws declaration  
• Surround with try/catch  
Press 'F2' for focus

Surround with try/catch 선택



# 일반 예외

## ● 일반 예외 – 컴파일러 체크

```
public class ExceptionHandling2 {  
  
    public static void main(String[] args) {  
        try {  
            Class.forName("java.lang.String2");  
            System.out.println("찾는 클래스가 있습니다.");  
        } catch (ClassNotFoundException e) {  
            System.out.println("클래스를 찾을 수 없습니다.");  
            e.printStackTrace();  
        }  
    }  
}
```

클래스를 찾을 수 없습니다.

```
java.lang.ClassNotFoundException: java.lang.String2  
    at java.base/jdk.internal.loader.BuiltinClassLoader.  
    at java.base/jdk.internal.loader.ClassLoaders$AppClas  
    at java.base/java.lang.ClassLoader.loadClass(ClassLoa  
    at java.base/java.lang.Class.forName0(Native Method)
```



# 일반 예외

## ● 숫자 입력시 문자 입력 예외 처리

```
public class GuessNumber {  
  
    public static void main(String[] args) {  
  
        Random rand = new Random();  
        int comNum = rand.nextInt(30) + 1; //1 ~ 30  
  
        Scanner sc = new Scanner(System.in);  
  
        while(true) {  
            try {  
                System.out.print("숫자(예: 1~30)를 입력: ");  
                int guessNum = sc.nextInt();  
  
                if(guessNum < 1 || guessNum > 30) {  
                    System.out.println("범위를 초과했어요. 다시 입력하세요");  
                }else {
```



# 일반 예외

## ● 숫자 입력시 문자 입력 예외 처리

```
        if(guessNum == comNum) {
            System.out.println("정답!");
            break;
        }else if(guessNum > comNum) {
            System.out.println("너무 커요!");
        }else {
            System.out.println("너무 작아요!");
        }
    }
} catch (InputMismatchException e) {
    System.out.println("유효한 숫자를 입력하세요");
    sc.nextLine(); //잘못된 입력 제거 - 버퍼 정리
}
}
sc.close();
}
```

숫자(예: 1~30)를 입력: 20  
너무 커요!  
숫자(예: 1~30)를 입력: 10  
너무 커요!  
숫자(예: 1~30)를 입력: abc  
유효한 숫자를 입력하세요  
숫자(예: 1~30)를 입력: 6  
너무 작아요!



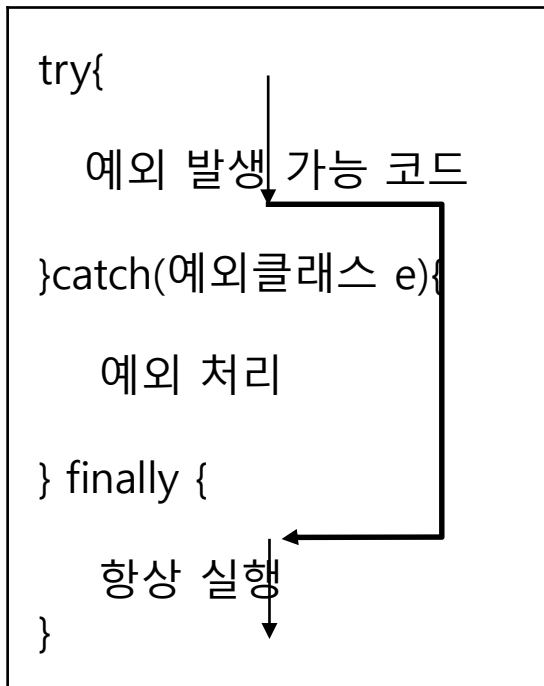
# try~catch~finally문

- try~catch~finally문 사용하기

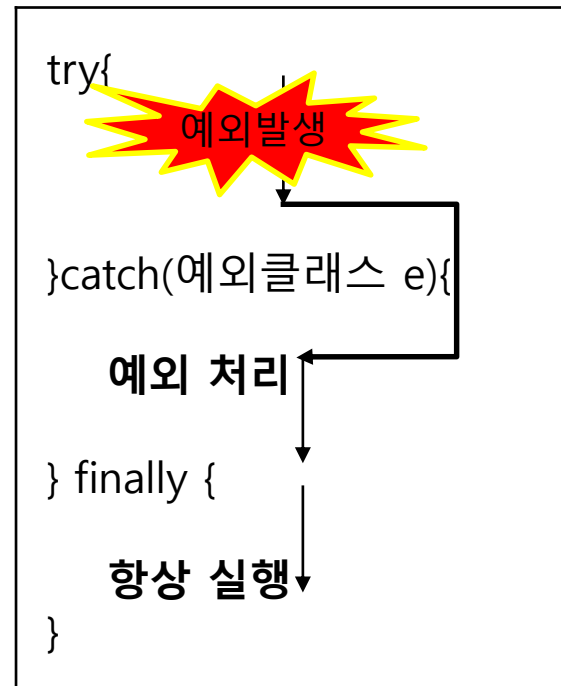
프로그램에서 외부장치와의 연동시 초기화나 마무리 작업시 주로 사용한다.

이때 사용하는 블록이 finally인데 일단 try블록이 수행되면 어떤 경우에도 반드시 수행된다.

정상실행 되었을 경우



예외가 발생되었을 경우



# try~catch~finally문

## ● try~catch~finally문 사용하기

```
System.out.println("=====");

try {
    Class.forName("java.lang.String2");
    System.out.println("찾는 클래스가 있습니다.");
} catch (ClassNotFoundException e) {
    System.out.println("클래스를 찾을 수 없습니다.");
    e.printStackTrace();
} finally {
    System.out.println("항상 수행됨");
}
}
```

클래스를 찾을 수 없습니다.

```
java.lang.ClassNotFoundException: java.lang.String2
    at java.base/jdk.internal.loader.BuiltinClass
    at java.base/jdk.internal.loader.ClassLoaders
    at java.base/java.lang.ClassLoader.loadClass(
    at java.base/java.lang.Class.forName0(Native
    at java.base/java.lang.Class.forName(Class.ja
    at java.base/java.lang.Class.forName(Class.ja
    at Chapter8/exceptions.ExceptionHandling3.mai
```

항상 수행됨



# 다중 예외 처리

- 다중 예외 – 동시에 여러 개의 예외가 발생한 경우

```
public class MultiException {  
  
    public static void main(String[] args) {  
  
        String[] array = {"100", "123a"};  
  
        for(int i = 0; i <= array.length; i++) {  
  
            System.out.println(array[i]); //배열 출력  
            //문자열을 정수로 변환  
            int value = Integer.parseInt(array[i]);  
            System.out.println(value);  
  
        }  
    }  
}
```





# 다중 예외 처리

## ● 다중 예외 처리

[ java.lang.ArrayIndexOutOfBoundsException ]

```
100
123a
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 2 out of bounds for length 2
    at Chapter8/exceptions.MultiException.main(MultiException.java:11)
```

[ java.lang.NumberFormatException ]

```
100
100
123a
Exception in thread "main" java.lang.NumberFormatException: For input string: "123a"
    at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
    at java.base/java.lang.Integer.parseInt(Integer.java:662)
    at java.base/java.lang.Integer.parseInt(Integer.java:778)
    at Chapter8/exceptions.MultiException.main(MultiException.java:13)
```



# 다중 예외 처리

## ● 다중 예외 처리

```
public class MultiException {  
    public static void main(String[] args) {  
        String[] array = {"100", "123a"};  
        for(int i = 0; i <= array.length; i++) {  
            try {  
                System.out.println(array[i]); //배열 출력  
                //문자열을 정수로 변환  
                int value = Integer.parseInt(array[i]);  
                System.out.println(value);  
            } catch (ArrayIndexOutOfBoundsException e) {  
                System.out.println("배열 인덱스가 초과되었습니다.");  
            } catch (NumberFormatException e) {  
                System.out.println("숫자로 변환할 수 없는 항목이 있습니다.");  
            }  
        }  
    }  
}
```

```
100  
100  
123a  
숫자로 변환할 수 없는 항목이 있습니다.  
배열 인덱스가 초과되었습니다.
```



# 예외 발생 시키기 – throw

## ● throw 로 예외 발생 시키기

**throw new**는 직접 예외를 발생시키는 기능을 한다.

**throw new 예외 클래스("메시지")**

-> 예외 객체를 생성하고 강제로 던짐

```
public class ThrowExample {  
    public static void checkAge(int age) {  
        if (age < 0) {  
            throw new IllegalArgumentException("나이는 음수가 될 수 없습니다.");  
        } else if (age < 20) {  
            throw new ArithmeticException("미성년자는 입장할 수 없습니다.");  
        } else {  
            System.out.println("입장 가능합니다.");  
        }  
    }  
}
```



# 예외 발생 시키기 – throw

- throw 로 예외 발생 시키기

```
public static void main(String[] args) {  
    try {  
        checkAge(10);  
    } catch (IllegalArgumentException e) {  
        System.out.println("예외 발생 " + e.getMessage());  
    } catch (ArithmeticException e) {  
        System.out.println("예외 발생 " + e.getMessage());  
    }  
}
```



# 예외 처리 미루기 – throws

- **throws** 로 예외처리 미루기(떠넘기기)

예외 처리를 해당 메서드에서 하지 않고 미룬 후, 메서드를 호출하여 사용하는 곳에서 예외를 처리하는 방법이다.

메서드명 **throws** 예외클래스1, 예외클래스2,..{  
}

```
public static void findClass() {  
    Class.forName("java.lang.Math");  
}
```

Add throws declaration 선택

Unhandled exception type: ClassNotFoundException  
2 quick fixes available:  
1. Add throws declaration  
2. Surround with try/catch  
Press 'F2' for focus



# 예외 처리 미루기 – throws

- throws 로 예외처리 미루기(떠넘기기)

```
public class ThrowsExample {  
  
    public static void main(String[] args) {  
        //호출(사용)한 곳에서 예외 처리함  
        try {  
            findClass();  
        } catch (ClassNotFoundException e) {  
            //e.printStackTrace();  
            System.out.println("예외 처리: " + e.toString());  
        }  
    }  
  
    public static void findClass() throws ClassNotFoundException {  
        //예외를 미룸  
        Class.forName("java.lang.Math2");  
    }  
}
```

예외 처리: [java.lang.ClassNotFoundException](#): java.lang.Math2

