

# Lab 12

Q: 관리자 클래스의 회원 아이디, 비밀번호 출력 void 함수와 겹치게 되는 것 같은데 그러면 (전체 회원을 출력하는) void 함수 내부에서 toString을 이용해야하는 것인가요?

A: toString()이 호출되는 부분이 아래에 있습니다. toString()은 Java의 모든 클래스의 최상위 클래스인 Object가 가지고 있는 메서드입니다.

```
public void printAllUsers() {  
    System.out.println("현재 등록된 모든 회원들의 정보입니다.");  
    for(User user : users) {  
        if(!(user instanceof Admin))  
            System.out.println(user);  
    }  
    System.out.println("전체 회원 수 : " + (users.size() - 1));  
}
```

Q: (메소드 설명), 이는 앞의 두 메소드를 통해 구현이 가능할 것 같습니다.  
혹시 없어도 구현이 가능하다면 포함시키지 않아도 되는지 궁금합니다.


Q: 관리자 class에 전체 회원 Array List를 만든 후, 로그인 프로그램에서는 전체회원리스트 부분을 구현하지 않고, 관리자 class에 있는 Array List에 접근하게 코드를 작성해도 될까요?

Q: 메뉴를 출력하고 선택하는 로직의 중복을 피하기 위해 클래스를 새로 만들려고 합니다.

A: 본인 코딩 스타일에 따라서 구현이 다르게 됩니다.  
문서는 guide로 참고해 주시고 핵심적인 내용을 잘 구현하시면 됩니다.

Q: 정규표현식(java.util.regex 패키지) 등의 라이브러리를 이용해도 될까요? 된다면 어느 정도의 범위까지 허용되는지 궁금합니다.

A: 라이브러리 이용은 자유롭게 해도 됩니다.



Q&A

<https://padlet.com/dcslabcp/0524-szvffj0wkmoq161m>

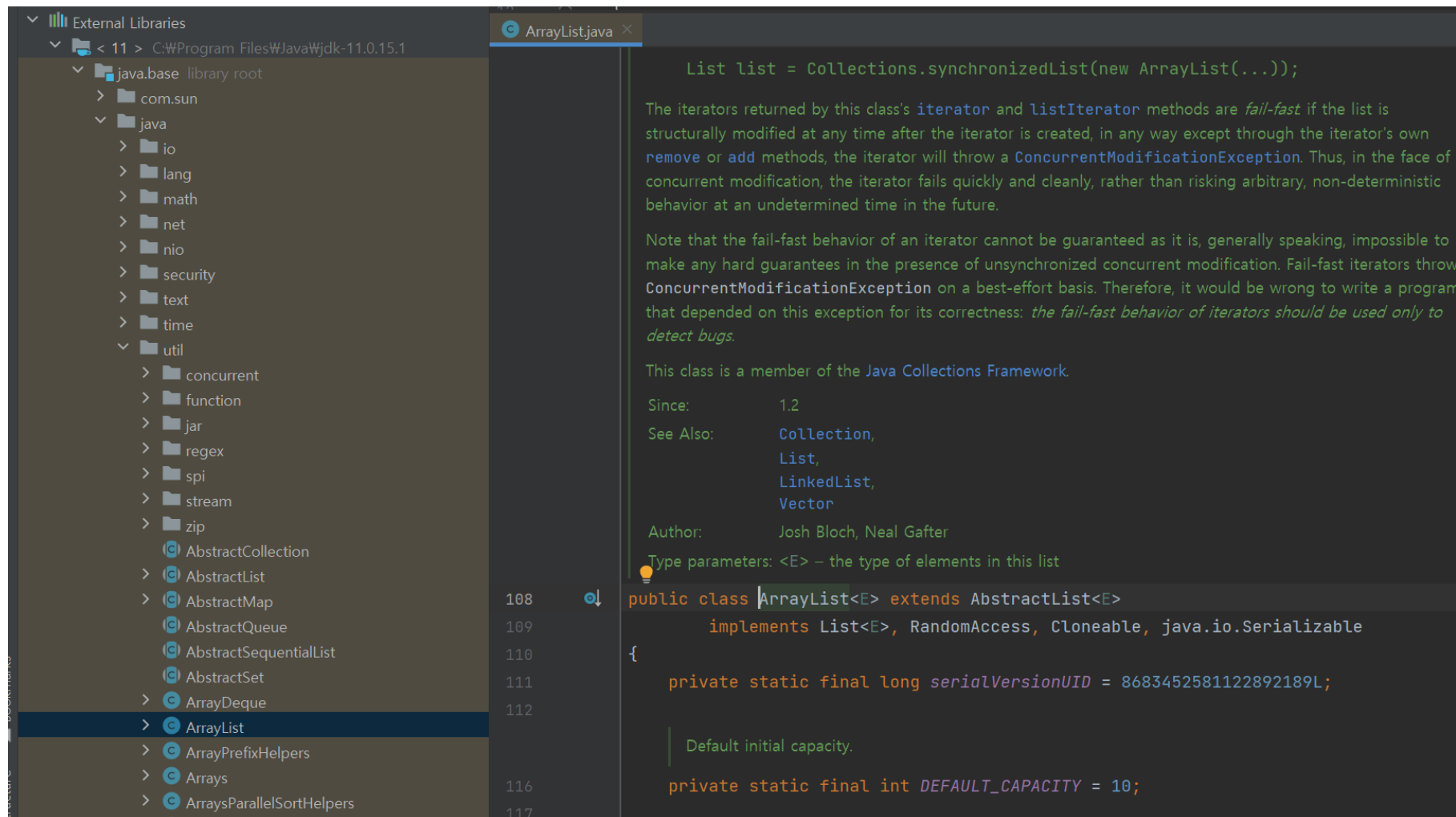
# Java library, package, module

- library : 개발자가 편리하게 사용할 수 있도록 패키지 혹은 모듈을 압축한 파일
  - package : 상호 관련 있는 클래스와 인터페이스를 한곳에 묶어 놓은 것
  - module : 밀접한 관계가 있는 패키지와 리소스를 묶어 놓은 것
- JDK를 설치하면 jmods 폴더에 jmod 파일을 제공하는데 jmod 파일이 모듈 파일

로컬 디스크 (C:) > Program Files > Java > jdk-11.0.15.1 > jmods

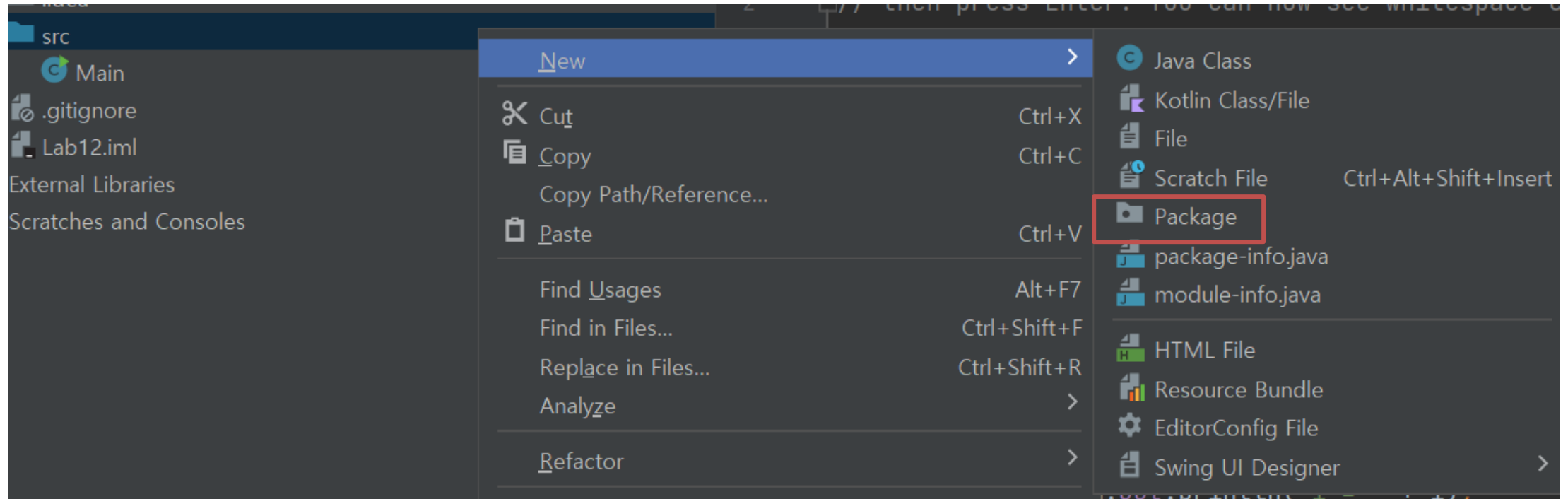
The screenshot shows the file system structure of the Java JDK 11.0.15.1 jmods directory. On the left, a list of jmod files is shown, all dated 2022-05-27 오후 7:18. The files include java.base.jmod, java.compiler.jmod, java.datatransfer.jmod, java.desktop.jmod, java.instrument.jmod, java.logging.jmod, java.management.jmod, java.management.rmi.jmod, java.naming.jmod, java.net.http.jmod, java.prefs.jmod, java.rmi.jmod, java.scripting.jmod, java.se.jmod, java.security.jgss.jmod, java.security.sasl.jmod, and java.smartcardio.jmod. On the right, the source code of the ArrayList class is displayed. The code shows the class ArrayList<E> extending AbstractList<E> and implementing List<E>, RandomAccess, Cloneable, and java.io.Serializable. It includes a private static final long serialVersionUID and a private static final int DEFAULT\_CAPACITY = 10. The code also shows the constructor and the add method.

# IntelliJ에서 확인하기



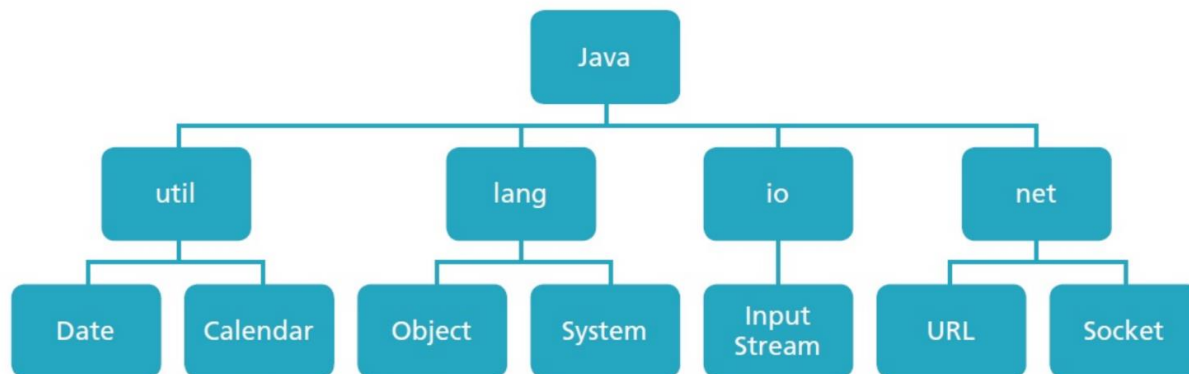


# Java package 생성



# Java의 주요 library, package, module

패키지	설명
java.awt	그래픽을 처리하는 API
java.io	입출력을 스트림하는 API
java.lang	자바 프로그램의 필수 API
java.math	수학과 관련된 API
java.net	네트워크를 처리하는 API
java.text	날짜, 시간, 문자열 등 지역화를 처리하는 API
java.time	JDK 8이 지원하는 날짜 및 시간을 처리하는 API
java.util	날짜, 리스트, 벡터 등 유틸리티 API
javax.swing	스윙 컴포넌트 API





# java.util 패키지

- java.util 패키지가 제공하는 주요 클래스

클래스	설명
Arrays	배열을 비교, 복사, 정렬 등 조작할 때 사용한다.
Calendar	날짜와 시간 정보가 필요할 때 사용한다.
Date	밀리초 단위의 현재 시각이 필요할 때 사용한다.
StringTokenizer	특정 문자로 구분된 문자열을 뽑아낼 때 사용한다.
Random	난수가 필요할 때 사용한다.

# java.util 패키지 - StringTokenizer

- 문자열을 토큰으로 분리하는 데 사용
- 공백이나 줄 바꿈 등 구분자를 사용해 문자열을 분리
- 주요 생성자

생성자	설명
<code>StringTokenizer(String s)</code>	주어진 문자열을 기본 구분자로 파싱한 <code>StringTokenizer</code> 객체를 생성한다.
<code>StringTokenizer(String s, String delim)</code>	주어진 문자열을 <code>delim</code> 구분자로 파싱한 <code>StringTokenizer</code> 객체를 생성한다.

- 기본 구분자는 공백, `\t`, `\n`, `\r`
- 주요 메서드

메서드	설명
<code>int countTokens()</code>	남아 있는 토큰의 개수를 반환한다.
<code>boolean hasMoreTokens()</code>	남아 있는 토큰이 있는지 여부를 반환한다.
<code>String nextToken()</code>	다음 토큰을 꺼내 온다.

# java.util 패키지 - StringTokenizer

- 지난 주 ip 파싱 문제를 StringTokenizer를 이용해 해결하면

```
import java.util.StringTokenizer;

public class StringTokenizerTest {
    public static void main(String[] args) {
        String ip_addr = "147.46.100.100";
        StringTokenizer st = new StringTokenizer(ip_addr, " ");
        while(st.hasMoreElements()) {
            System.out.print(st.nextToken() + " ");
        }
    }
}
```

147 46 100 100

# 연습문제 1

- input.txt파일을 읽어와서 Data를 이용한 ArrayList 출력 결과가 나오는 코드 작성

```
class Data {  
    2 usages  
    private String ip;  
    2 usages  
    private String port;  
    2 usages  
    private int id;  
  
    1 usage  
    public Data(String ip, String port, int id) {  
        this.ip = ip;  
        this.port = port;  
        this.id = id;  
    }  
    public String toString() {  
        return "id : " + ip + " port : " + port + " id : " + id;  
    }  
}
```

```
[id : 15.165.44.218 port : 27000 id : 0, id : 3.34.185.220 port : 27000 id : 0, id : 3.38.134.55 port : 27000 id : 0, id : 13.124.17.213 port : 27000 id : 0]
```

```

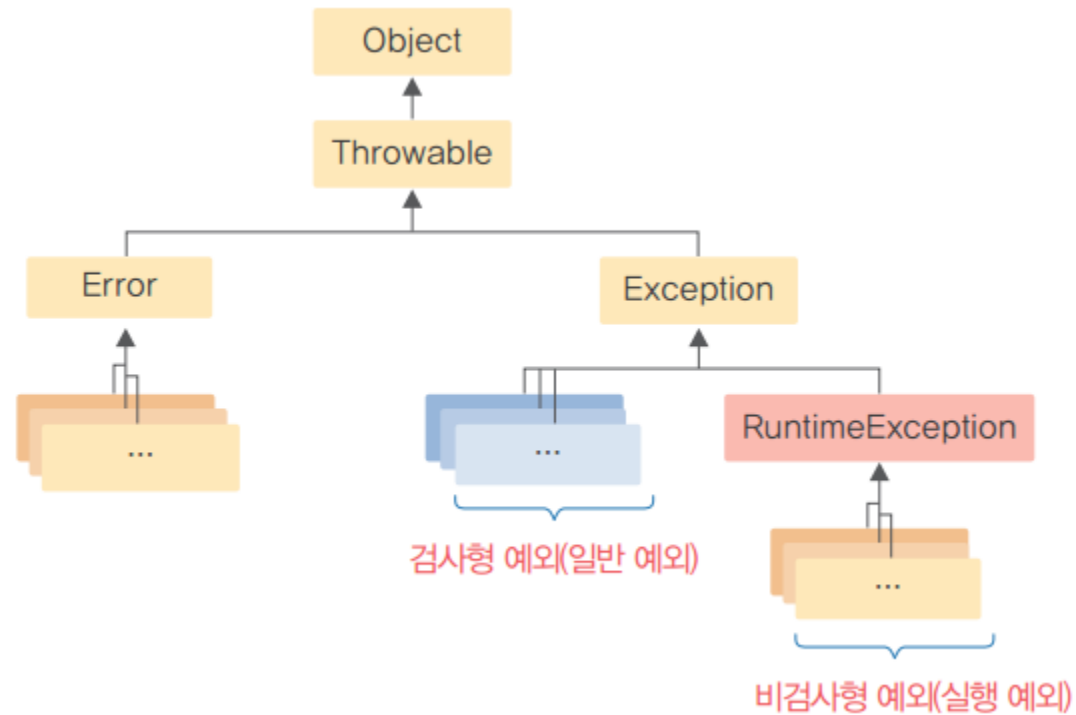
public class Practice1 {
    public static void main(String[] args) {
        ArrayList<Data> dataArrayList = new ArrayList<>();
        try {
            BufferedReader br = new BufferedReader(new FileReader("src/input.txt"));
            String line;
            while((line = br.readLine()) != null) {
                StringTokenizer st = new StringTokenizer(line, ",");
                String ip = st.nextToken();
                String port = st.nextToken();
                String id = st.nextToken();
                id = id.substring("user".length(), id.length());
                dataArrayList.add(new Data(ip, port, Integer.parseInt(id)));
            }
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

        System.out.println(dataArrayList);
    }
}

```

# Exception

- 잘못된 코드, 부정확한 데이터, 예외적인 상황에 의하여 발생하는 오류를 적절히 처리하는 기법
- 일반 예외와 실행 예외로 구분할 수 있음





# Exception

- 실행 예외 : JVM은 해당하는 실행 예외 객체를 생성
- 실행 예외는 컴파일러가 예외 처리 여부를 확인하지 않음.  
따라서 개발자가 예외 처리 코드의 추가 여부를 결정

실행 예외	발생 이유
ArithmeticException	0으로 나누기와 같은 부적절한 산술 연산을 수행할 때 발생한다.
IllegalArgumentException	메서드에 부적절한 인수를 전달할 때 발생한다.
IndexOutOfBoundsException	배열, 벡터 등에서 범위를 벗어난 인덱스를 사용할 때 발생한다.
NoSuchElementException	요구한 원소가 없을 때 발생한다.
NullPointerException	null 값을 가진 참조 변수에 접근할 때 발생한다.
NumberFormatException	숫자로 바꿀 수 없는 문자열을 숫자로 변환하려 할 때 발생한다.

# Exception

- 일반 예외 : 컴파일러는 발생할 가능성을 발견하면 컴파일 오류를 발생
- 개발자는 예외 처리 코드를 반드시 추가

일반 예외	발생 이유
ClassNotFoundException	존재하지 않는 클래스를 사용하려고 할 때 발생한다.
InterruptedException	인터럽트되었을 때 발생한다.
NoSuchFieldException	클래스가 명시한 필드를 포함하지 않을 때 발생한다.
NoSuchMethodException	클래스가 명시한 메서드를 포함하지 않을 때 발생한다.
IOException	데이터 읽기 같은 입출력 문제가 있을 때 발생한다.

# Exception 처리

- try 블록이 제일 먼저 나옴
- catch는 0개 이상 (매개 변수는 다 달라야 함)
- finally는 0개 or 1개만 가능
- finally는 예외가 발생하지 않아도 수행

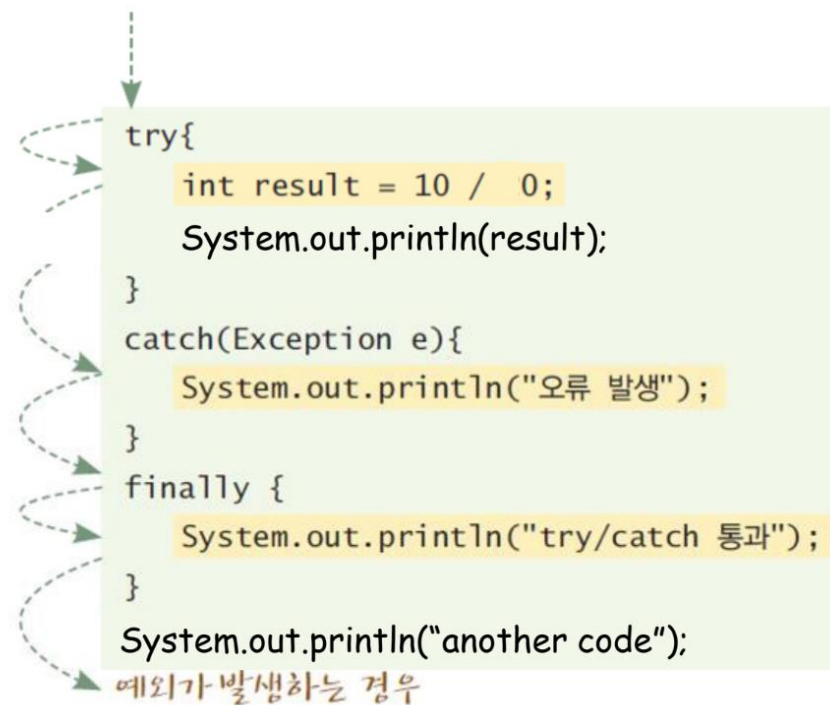
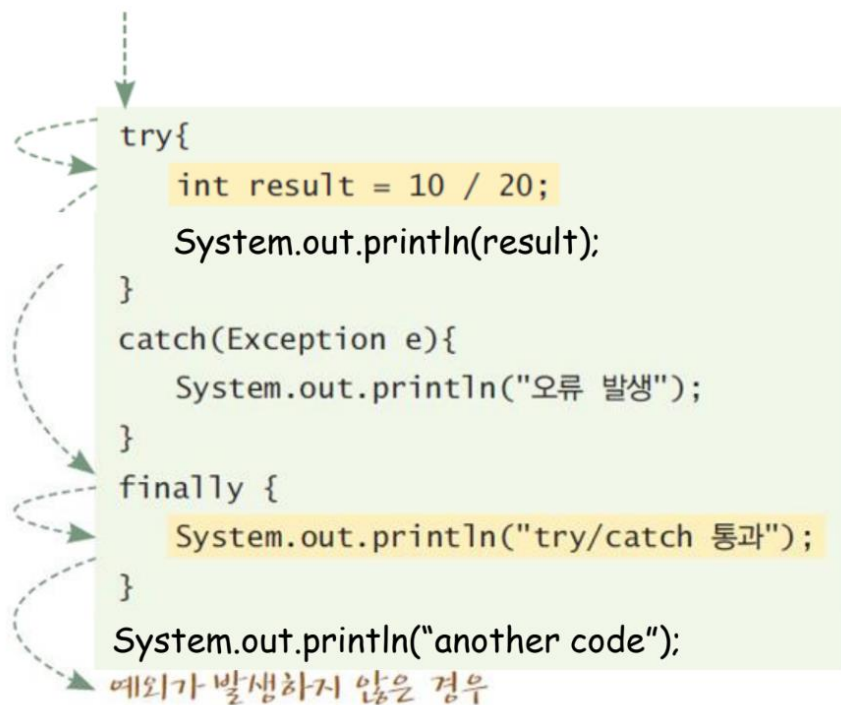
```
try {  
    예외가 발생할 수 있는 실행문;  
} catch (예외 클래스1 | 예외 클래스2 변수) {  
    핸들러;  
} catch (예외 클래스3 변수) {  
    핸들러;  
} finally {  
    예외 발생 여부와 관계없이 수행할 실행문;  
}
```

다수의 예외를 한꺼번에 잡으려면 | 연산자로 연결하면 된다.

여러 개의 catch 블록이 있을 수 있다.

없어도 상관없다. 있다면 예외 발생 여부와 관계없이 실행된다.

# try/catch 실행 흐름



[try/catch 블록에서의 실행 흐름]

# 연습문제 2

- 주어진 배열을 이용하여 6번째 요소에 접근을 시도하면 아래와 같은 출력이 되도록 try/catch문을 작성해보자

```
int[] array = {1, 2, 3, 4, 5};
```

- 출력 결과

```
1 2 3 4 5 인덱스 범위를 초과했습니다.
```

```
public class Practice2 {  
    public static void main(String[] args) {  
        int[] array = {1, 2, 3, 4, 5};  
  
        try {  
            for(int i = 0; i <= array.length; i++)  
                System.out.print(array[i] + " ");  
        } catch (ArrayIndexOutOfBoundsException e) {  
            System.out.println("인덱스 범위를 초과했습니다.");  
        }  
    }  
}
```



# 연습문제 3

- 두 숫자를 입력 받아 첫 번째 숫자에서 두 번째 숫자를 나누는 프로그램을 작성하자.
- 숫자가 아닌 입력이 들어왔을 때 발생할 수 있는 예외를 처리하자.
- 0으로 나눴을 때 발생할 수 있는 예외를 처리하자.

- 실행1

```
나누기를 수행할 두 숫자를 공백을 두고 입력하세요 : aaa bbb
숫자를 입력하셔야 합니다.
```

- 실행2

```
나누기를 수행할 두 숫자를 공백을 두고 입력하세요 : 10 0
0으로 나눌 수 없습니다.
```

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class Practice3 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("나누기를 수행할 두 숫자를 공백을 두고 입력하세요 : ");

        double result = 0;
        try {
            int n1 = sc.nextInt();
            int n2 = sc.nextInt();
            // double result;
            result = n1 / n2;
        } catch (InputMismatchException e) {
            System.out.println("숫자를 입력하셔야 합니다.");
        } catch (ArithmeticException e) {
            System.out.println("0으로 나눌 수 없습니다.");
        } finally {
            sc.close();
        }
    }
}
```

# Quiz

- 이 프로그램의 실행을 예측해 봅시다.

```
try {  
    ...  
} catch (Exception e) {  
    ...  
} catch (InputMismatchException e) {  
    ...  
}
```

# Quiz

```
try {  
    ...  
} catch (Exception e) {  
    ...  
} catch (InputMismatchException e) {  
    ...  
}
```

Compile Error

- 여러 개의 catch 블록 사용 시
- 첫 번째 catch 블록부터 match 여부 조사
- 발생한 예외와 타입이 맞지 않으면 계속 아래로 감
- 타입이 일치하는 블록을 만나면 그 이후에 나오는 catch 블록은 더 이상 조사 X

# Exception 처리 2

- 메서드에서 발생한 예외를 내부에서 처리하기 부담스러울 경우 throws 키워드 사용해 예외를 상위 코드 블록으로 양도 가능

```
public void write(String filename)
    throws IOException, ReflectiveOperationException {
    // 파일 쓰기과 관련된 실행문 ...
}
```

throws는 예외를 다른 메서드로 떠넘기는 키워드이다.

예외를 1개 이상 선언할 수 있다.

# Throws를 사용하는 예

```
class Test {  
    1 usage  
    public void add (String x, String y) throws NumberFormatException {  
        int result = Integer.parseInt(x) + Integer.parseInt(y);  
        System.out.println("결과는 " + result + "입니다.");  
    }  
}  
  
public class ThrowsTest {  
    public static void main(String[] args) {  
        Test test = new Test();  
        try {  
            test.add(x: "123", y: "ㄱㄴㄷ");  
        } catch (NumberFormatException e) {  
            System.out.println("숫자를 넣어 주어야 합니다.");  
        }  
    }  
}
```



# Throwable의 주요 메서드

메서드	설명
<code>public String getMessage()</code>	Throwable 객체의 자세한 메시지를 반환한다.
<code>public String toString()</code>	Throwable 객체의 간단한 메시지를 반환한다.
<code>public void printStackTrace()</code>	Throwable 객체와 추적 정보를 콘솔 뷰에 출력한다.

```
Exception in thread "main" java.lang.RuntimeException Create breakpoint : java.io.FileNotFoundException: input.txt (지정된 파일을 찾을 수 없습니다)
    at ThrowableMethodTest.main(ThrowableMethodTest.java:14)
Caused by: java.io.FileNotFoundException Create breakpoint : input.txt (지정된 파일을 찾을 수 없습니다)
    at java.base/java.io.FileInputStream.open0(Native Method)
    at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
    at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
    at java.base/java.io.FileInputStream.<init>(FileInputStream.java:112)
    at java.base/java.io.FileReader.<init>(FileReader.java:60)
    at ThrowableMethodTest.main(ThrowableMethodTest.java:12)
```

# 사용자 정의 예외

- Exception 클래스의 하위 클래스인 사용자 정의 예외를 만들 수 있음
- Exception 클래스 생성자
  - Exception()
  - Exception(String message)
    - 매개변수로 전달된 문자열은 소속 메서드 getMessage()를 호출했을 때 반환됨

```
3 usages
class CustomException extends Exception {
    1 usage
    public CustomException() { super("내가 정의하는 예외"); }
}
```

# Exception 발생

```
public class CustomExceptionTest {  
    1 usage  
    public static void method1() throws CustomException {  
        throw new CustomException();  
    }  
    public static void main(String[] args) {  
        try {  
            method1();  
        } catch (CustomException e) {  
            System.out.println(e.getMessage());  
            e.printStackTrace();  
        }  
    }  
}
```

내가 정의하는 예외

CustomException: 내가 정의하는 예외

at CustomExceptionTest.method1(CustomExceptionTest.java:9)

at CustomExceptionTest.main(CustomExceptionTest.java:13)

# 연습문제 4

```
int[] array = new int[] {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

- 배열에서 해당 숫자(data)를 찾는 searchArray(int[] array, int data) 정적 메서드를 작성해보자
- 만약 배열에 원하는 숫자가 없으면 NotFoundException 예외(custom exception)를 발생한다.
- 실행1

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
배열에서 찾으려는 숫자를 입력하세요: 100
NotFoundException: 배열에서 찾을 수 없습니다.
    at Practice4.searchArray(Practice4.java:33)
    at Practice4.main(Practice4.java:22)
```

- 실행2

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
배열에서 찾으려는 숫자를 입력하세요: 10
데이터는 인덱스 9에 있습니다.
```

```

import java.util.Arrays;
import java.util.Scanner;

3 usages
class NotFoundException extends Exception {
    1 usage
    public NotFoundException () { super("배열에서 찾을 수 없습니다."); }
}

```

```

public class Practice4 {
    public static void main(String[] args) {
        int[] array = new int[] {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

        System.out.println(Arrays.toString(array));
        Scanner sc = new Scanner(System.in);
        System.out.print("배열에서 찾으려는 숫자를 입력하세요: ");
        int n = sc.nextInt();
        try {
            System.out.println("데이터는 인덱스 " + searchArray(array, n) + "에 있습니다.");
        } catch (NotFoundException e) {
            e.printStackTrace();
        }
    }
}

```

```

1 usage
public static int searchArray(int array[], int data) throws NotFoundException {
    for(int i = 0; i < array.length; i++ )
        if(array[i] == data)
            return i;
    throw new NotFoundException();
}

```

# 추상 클래스(abstract class)

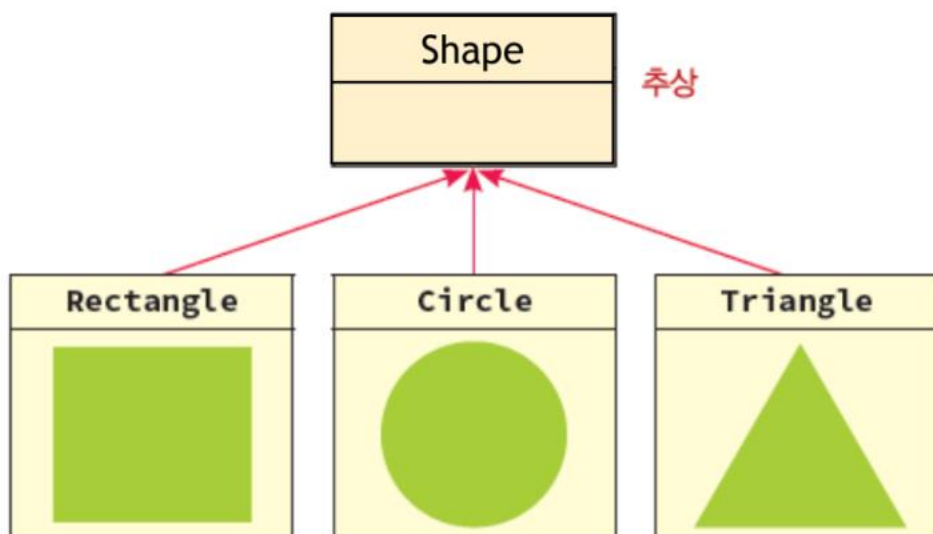
- 아래와 같이 정의해서 사용
- 추상 메서드를 1개라도 가지면 추상 클래스로 정의

```
1  abstract class Shape {  
    1 usage  
2      int x, y; // center  
    no usages  
3      public void move(int x, int y) {  
4          this.x = x;  
5          this.y = y;  
6      }  
    no usages  
7      public abstract double area();  
8  }
```



# 추상 클래스(abstract class) 사용

- 추상 클래스를 상속받은 하위 클래스에서는 모든 추상 메서드를 구현해야 함
  - 만약 모든 하위 클래스에서 추상 메서드를 구현하지 않았다면 어떻게 해야 할까?
  - 만약 아래와 같은 구조에서 Triangle이 area() 를 구현하지 않았다면?



# 연습 문제 5

```
public class Practice5 {  
    public static void main(String[] args) {  
        Rectangle r = new Rectangle();  
        Circle c = new Circle();  
        // Triangle t = new Triangle();  
        r.setWidth(5);  
        r.setHeight(3);  
        c.setRadius(4);  
  
        System.out.println("사각형 넓이 " + r.area());  
        System.out.println("원 넓이 " + c.area());  
    }  
}
```

- 왼쪽과 같은 main 함수가  
아래와 같이 실행하도록 작성해 봅시다.  
(Shape 클래스는 앞선 슬라이드 참조)

```
사각형 넓이 15.0  
원 넓이 50.26548245743669
```

```

class Rectangle extends Shape {
    2 usages
    private int width, height;

    2 usages
    @Override
    public double area() {
        return width * height;
    }

    1 usage
    public void setWidth(int width) { this.width = width; }

    1 usage
    public void setHeight(int height) { this.height = height; }
}

```

```

class Circle extends Shape {
    2 usages
    private int radius;
    2 usages
    @Override
    public double area() { return Math.pow(radius, 2) * Math.PI; }

    1 usage
    public void setRadius(int radius) { this.radius = radius; }
}

no usages
abstract class Triangle extends Shape {
    no usages
    int base;
    no usages
    int height;
    // no area method
}

```

# 인터페이스 (interface)

- 기본 구조

```
interface 인터페이스이름 {  
    // 상수 필드        → 상수만 가능하기 때문에 public static final 키워드 생략 가능  
  
    // abstract 메서드    → public abstract 키워드 생략 가능  
}
```

- 변수 필드는 선언 불가
- 보통 클래스와 마찬가지로 별도의 파일에 저장

# 인터페이스 (interface)

- 자바에서는 기본적으로 제공하는 인터페이스가 있음
- 대표적으로
  - java.lang 패키지 → CharSequence, Comparable, Runnable 등
  - java.util 패키지 → Collection, Comparator, List 등
- 객체의 크기를 비교하는 Comparable 인터페이스

```
public interface Comparable {
```

```
    int compareTo(Object other);
```

```
}
```

객체 other보다 크면 양수, 같으면 0,  
작으면 음수를 반환한다.

# Comparable 사용 예

```
public class Box implements Comparable{
    5 usages
    private double volume;
    2 usages
    public Box(double v) {
        volume = v;
    }
    @Override
    public int compareTo(Object o) {
        Box other = (Box) o;
        if(this.volume < other.volume) return -1;
        else if (this.volume > other.volume) return 1;
        else return 0;
    }

    public static void main(String[] args) {
        Box b1 = new Box( v: 100);
        Box b2 = new Box( v: 85.2);
        if(b1.compareTo(b2) > 0)
            System.out.println("b1이 더 큼니다.");
        else
            System.out.println("b1이 b2보다 작거나 같습니다.");
    }
}
```

b1이 더 큼니다.

# 연습문제 6

- Student 클래스는 이름, 학점 필드를 Comparable 인터페이스를 implements 한다.
  - Comparable에 정의된 compareTo 구현 시 학점이 낮은 순서대로 정렬하고
  - 동점자가 있으면 이름 가나다 순서대로 정렬한다.
- 다음과 같은 출력이 되도록 Student 클래스를 작성해보자

```
public class Practice6 {  
    public static void main(String[] args) {  
        Student[] students = new Student[5];  
        students[0] = new Student( name: "홍길동", gpa: 3.39);  
        students[1] = new Student( name: "김서울", gpa: 4.00);  
        students[2] = new Student( name: "최کم공", gpa: 3.70);  
        students[3] = new Student( name: "박철수", gpa: 3.70);  
        students[4] = new Student( name: "이영희", gpa: 3.39);  
  
        System.out.println(Arrays.toString(students));  
        Arrays.sort(students);  
        System.out.println(Arrays.toString(students));  
    }  
}
```

```
[이름 : 홍길동, 평점 : 3.39, 이름 : 김서울, 평점 : 4.0, 이름 : 최کم공, 평점 : 3.7, 이름 : 박철수, 평점 : 3.7, 이름 : 이영희, 평점 : 3.39]  
[이름 : 이영희, 평점 : 3.39, 이름 : 홍길동, 평점 : 3.39, 이름 : 박철수, 평점 : 3.7, 이름 : 최کم공, 평점 : 3.7, 이름 : 김서울, 평점 : 4.0]
```

```

import java.util.Arrays;
9 usages
class Student implements Comparable {
4 usages
    private String name;
6 usages
    private double gpa;

5 usages
    public Student(String name, double gpa) {
        this.name = name;
        this.gpa = gpa;
    }
    public String toString() {
        return "이름 : " + name + ", 평점 : " + gpa;
    }

    @Override
    public int compareTo(Object o) {
        Student other = (Student) o;
        if(this.gpa < other.gpa) return -1;
        else if (this.gpa == other.gpa) {
            if(this.name.compareTo(other.name) < 0) return -1;
            else return 1;
        } else return 1;
    }
}

```