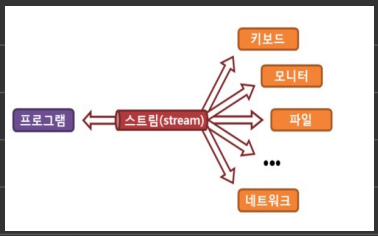
Stream

자바에서 파일이나 콘솔의 입출력은 직접 다루지 않고, 스트림이라는 흐름을통해 다룬다.

스트림이란 쉽게 입력이나 출력이 표현된 데이터의 이상화된 흐름을 의미.

즉, 스트림은 운영체제에 의해 생성되는 가상의 연결고리, 중간 매개자 역할

JAVA SE 8 의 스트림 API 와 다른개념이다.



JAVA.IO 패키지를 통해 InputStream과 OutputStream 클래스제공.

→ 스트림 생성이란 이러한 스트림 클래스 타입의 인스턴스를 생성 한다는 의미.

- InputStream
  read() 메소드
- OutputStream
  Write() 메소드.

클래스	메소드	설명
InputStream	abstract int read()	해당 입력 스트림으로부터 다음 바이트를 읽어들임.
	int read(byte[] b)	해당 입력 스트림으로부터 특정 바이트를 읽어들인 후, 배열 b에 저장함.
	int read(byte[] b, int off, int len)	해당 입력 스트림으로부터 len 바이트를 읽어들인 후, 배열 b[off]부터 저장함.
OutputStream	abstract void write(int b)	해당 출력 스트림에 특정 바이트를 저장함.
	void write(byte[] b)	배열 b의 특정 바이트를 배열 b의 길이만큼 해당 출력 스트림에 저장함.
	void write(byte[] b, int off, int len)	배열 b[off]부터 len 바이트를 해당 출력 스트림에 저장함.

read() 메소드는 해당 입력 스트림에서 더 이상 읽어들일 바이트가 없으면, -1을 반환해야 합니다.
그런데 반환 타입을 byte 타입으로 하면, 0부터 255까지의 바이트 정보는 표현할 수 있지만 -1은 표현할 수 없게 됩니다.
따라서 InputStream의 read() 메소드는 반환 타입을 int형으로 선언하고 있습니다.

Byte based Stream

스트림은 기본적으로 바이트 단위로 데이터를 전송.

입력 스트림	출력 스트림	입출력 대상
FileInputStream	FileOutputStream	파일
ByteArrayInputStream	ByteArrayOutputStream	메모리
PipedInputStream	PipedOutputStream	프로세스
AudioInputStream	AudioOutputStream	오디오 장치

보조스트림

실제로 데이터를 주고받는 역할은 없지만, 다른스트림의 기능을향상/새로운기능

추가해주는 스트림.

입력 스트림	출력 스트림	설명
FilterInputStream	FilterOutputStream	필터를 이용한 입출력
BufferedInputStream	BufferedOutputStream	버퍼를 이용한 입출력
DataInputStream	DataOutputStream	입출력 스트림으로부터 자바의 기본 타입으로 데이터를 읽어올 수 있게 함.
ObjectInputStream	ObjectOutputStream	데이터를 객체 단위로 읽거나, 읽어 들인 객체를 역직렬화시킴.
SequenceInputStream	X	두 개의 입력 스트림을 논리적으로 연결함.
PushbackInputStream	X	다른 입력 스트림에 버퍼를 이용하여 push back이나 unread와 같은 기능을 추가함.
X	PrintStream	다른 출력 스트림에 버퍼를 이용하여 다양한 데이터를 출력하기 위한 기능을 추가함.

문자기반 스트림

Char 형은 2byte → 1byte 전용하는 스트림 처리↓

주로 두 Stream은 Reader/Writer로 변경하여 사용

입력 스트림	출력 스트림	입출력 대상
FileReader	FileWriter	파일
CharArrayReader	CharArrayWriter	메모리
PipedReader	PipedWriter	프로세스
StringReader	StringWriter	문자열

입력 스트림	출력 스트림	설명
FilterReader	FilterWriter	필터를 이용한 입출력
BufferedReader	BufferedWriter	버퍼를 이용한 입출력
PushbackReader	X	다른 입력 스트림에 버퍼를 이용하여 push back이나 unread와 같은 기능을 추가함.
X	PrintWriter	다른 출력 스트림에 버퍼를 이용하여 다양한 데이터를 출력하기 위한 기능을 추가함.

자바 스트림 API

```
public class StreamDemo {

    public static void main(String[] args) {
        String[] strArr = { "aaa", "ddd", "ccc" };
        List<String> strList = Arrays.asList(strArr);

        Stream<String> strStream1 = strList.stream();
        Stream<String> strStream2 = Arrays.stream(strArr);

        strStream1.sorted().forEach(System.out::println);
        strStream2.sorted().forEach(System.out::println);
    }

}
```

type

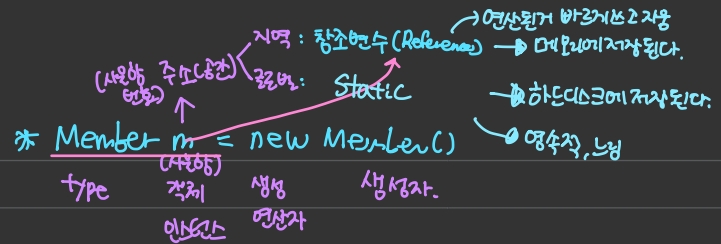
```
    public static void main(String[] args) {

        Arrays.asList("aaa", "ddd",
"ccc").stream().sorted().forEach(System.out::println);

    }
```

스트림으로 변환 → 컨슈머가 온 : 마지막은 컨슈머로 끝내야한다

```
Arrays.asList("a").stream().sorted().collect(Collectors.toList());

public class StreamMain {

    public static void main(String[] args) {

        new StreamUtil().arrayToList(new String[] { "a", "b",
"c" }).forEach(System.out::println);
    }

}
```

리스트로 리턴하는 구조

메소드참조, 여기서 타입은 보이드이다



메서드참조(P.566)

React's useRef()

메서드를 객체화

⇒ 무상태 : 메모리에 상태를 올리지 않아서. → 없는 상태

slice

프랙탈(영어: fractal) 또는 프랙털은 일부 작은 조각이 전체와 비슷한 기하학적 형태를 말한다. 이런 .특징을 자기 유사성이라고 하며, 다시 말해 자기 유사성을 갖는 기하학적 구조를 프랙탈 구조라고 한다. 브누아 망델브로가 처음으로 쓴 단어로, 어원은 조각났다는 뜻의 라틴어 형용사 'fractus'이다. 프랙탈 구조는 자연물에서 뿐만 아니라 수학적 분석, 생태학적 계산, 위상 공간에 나타나는 운동모형 등 곳곳에서도 발견되어 자연이 가지는 기본적인 구조이다. 불규칙하며 혼란스러워 보이는 현상을 배후에서 지배하는 규칙도 찾아낼 수 있다. 복잡성의 과학은 이제까지의 과학이 이해하지 못했던 불규칙적인 자연의 복잡성을 연구하여 그 안의 숨은 질서를 찾아내는 학문으로, 복잡성의 과학을 대표하는 혼돈 이론에도 프랙탈로 표현될 수 있는 질서가 나타난다.

프랙탈은 수학적 도형으로도 연구되고 있다. 프랙탈 도형은 종종 컴퓨터 소프트웨어를 이용한 재귀적이거나 반복적인 작업에 의한 반복되는 패턴으로 만들어진다. 대표적인 프랙탈 도형에는 망델브로 집합, 칸토어 집합, 시에르핀스키 삼각형, 페아노 곡선, 코흐 곡선 등이 있다. 프랙탈은 결정론적이거나 추계학적일 수 있으며, 혼돈적 계와 연관지어 발생할 수도 있다.

프랙탈 기하학은 프랙탈의 성질을 연구하는 수학 분야의 하나이다. 이는 과학, 공학, 컴퓨터 예술에 적용되기도 한다. 자연계에서도 프랙탈 구조가 자주 발견되며, 구름, 산, 번개, 난류, 해안선 및 나뭇가지 등이 여기에 해당한다. 프랙탈은 실용적인 목적으로 많이 사용되며, 현실 세계의 매우 불규칙한 물체들을 표현하기 위해서 쓰일 수 있다. 프랙탈 기법은 과학의 여러 분야에서는 물론, 기술적으로 이미지 압축 등에서도 사용된다.

Member m = new Member()

타입 객체 = new 생성자

↓커서백

타입변수 0

객체 = 레퍼런스 변수

클래스 내부변수 = 인스턴스의 변수

클래스 static 변수 = 클래스의 변수

class Box<T> { }

↓ ↓

변수명X

Type

Default

Type Valuable

Comparable vs Comparator

1. Interface Comparable.

정렬수행시 (Sorted()) 기본적으로 적용되는 정렬 기준이 되는 메서드를 정의한 인터페이스

- 정렬할 객체에 implements Comparable<T> 한후 compareTo 구현.

현재객체 < 파라미터 객체 ⇒ 음수 → 자리유지
" == " ⇒ 0 →
" > " ⇒ 양수 → 두객체 자리가 변경.

2. Interface Comparator

정렬 가능한 클래스들의 기본정렬과 다르게 정렬 하고 싶은때.

- 익명 클래스로 사용 / 내림. 오른차순 정렬시 사용.
- implements Comparator 후 compare() 메소드를 구현.

오름차순: Integer.compare(x,y) return (x>y)? 1 :((x==y)? 0: -1);

내림차순: Integer.compare(y, x)