Java 8 Stream

1. Stream 이란?

1. Stream - 사람・차량들로 계속 이어진 줄

> 데이터가 연속된 열의 구조로 되어 있는 형태
>
> 배열도 일종의 스트림이라고 할 수 있다

2. Streaming ?

> 스트리밍은 스트림이 끊이지 않고 계속되는 데이터 전송 방식

예) 동영상스트리밍
동영상데이터가 일정한 단위로 연속적으로 전송되는 방식

2. Java 에서의 Stream

1. Java Stream

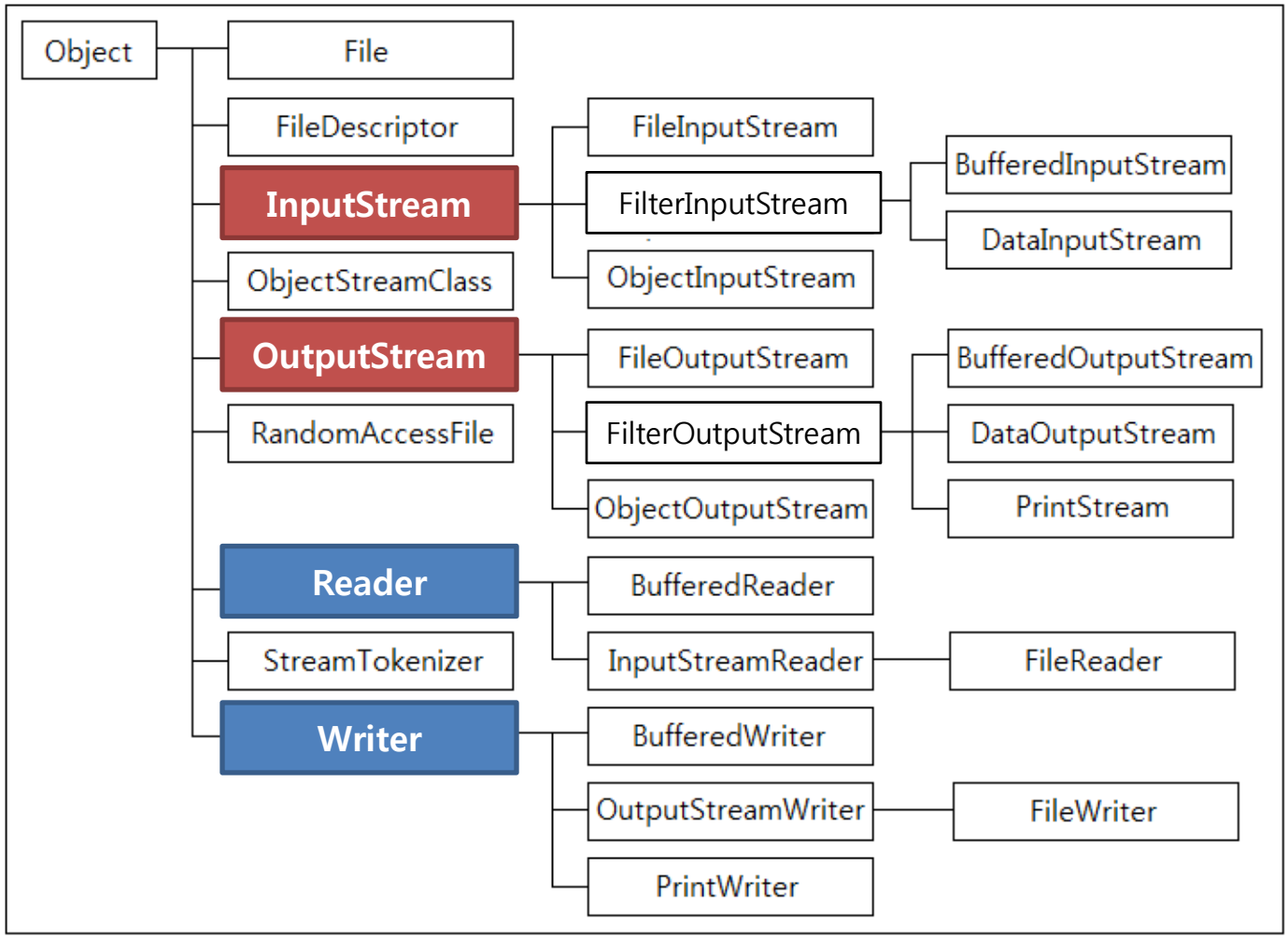
입출력을 처리하기 위한 표준 API

- Java 의 스트림은 다양한 입출력을 처리하기위한 중간매개체이다

- 다양한 입출력 종류
  a. 파일 < 디스크
  b. 키입력 데이터 < 키보드
  c. 마우스 포인터 < 마우스
  d. 화면출력 > 모니터
  e. 데이터전송 > 네트워크

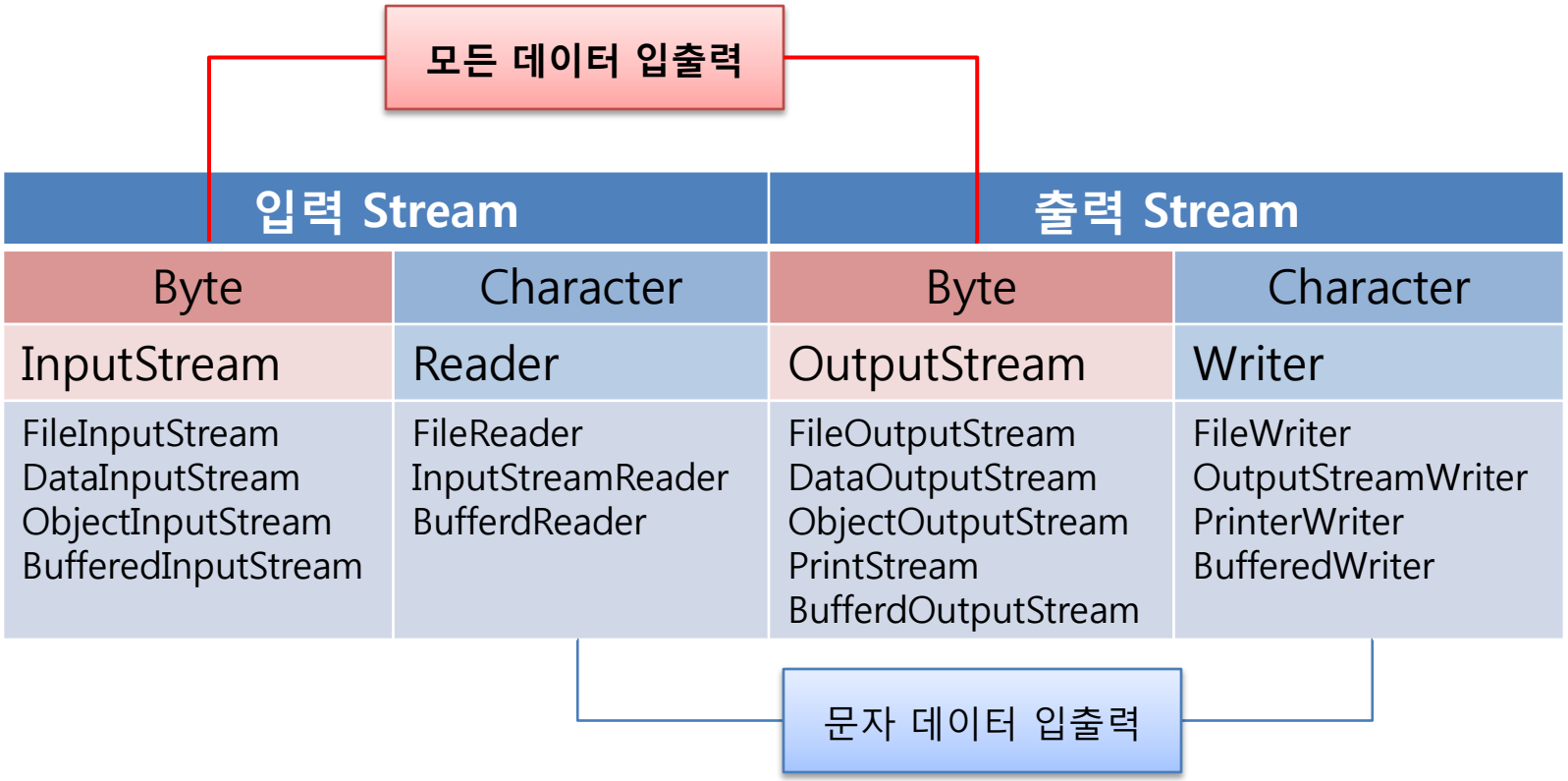
- 위와같이 다양한 입출력 장치에 대해 표준화된 입출력 도구가 Java의 Stream 이다

3. Java 에서의 IO 패키지에서의 Stream



4. Java 의 Stream 구성

- 아래 표와 같이 Stream 은 크게 입력, 출력 두가지 형태로 구분할 수 있는데 Byte 기반 스트림은 이미지, 동영상, 문자 등 모든 종류의 미디어를 주고받을 수 있는데 반해, Character 기반 스트림은 문자만 주고 받도록 설계되어 있다



5. Java 입출력 Stream 예제

1. 파일 입출력 스트림 - 파일 읽고 쓰기

```
// 입력 스트림 생성 < image.jpg 파일을 읽어온다
InputStream fis = new FileInputStream("image.jpg");

// 출력 스트림 생성 > new_image.jpg 파일로 출력(생성)한다
OutputSteam fos = new FileOutputStream("new_image.jpg");

int readCount; // 읽어온 바이트의 개수
byte[ ] buffer = new byte[1024];
while((readCount = fis.read(buffer)) != -1){ //buffer단위로 읽는다
      // buffer 단위로 파일을 출력한다
      fos.write(buffer, 0, readCount);
}
fis.close(); // 입력과 출력 Stream 을 닫아서 자원을 반환한다
fos.close();
```


6. Java8에서의 Stream 클래스

- 객체배열인 컬렉션을 연속된 데이터의 열로 사용할 수 있게 해주는 신규 내장객체로써 아래와 같은 특징이 있다

1. List 등의 객체배열에서 .stream의 형태로 추출할 수 있다

2. 람다코드를 적용하여 코드량을 줄일 수 있다.

3. 내부 반복자 API를 가지고 있다

4. 데이터의 병렬처리가 가능하다

```
Stream<T> stream = list.parallelStream( );
stream.forEach( a -> System.out.println( a ) );
```

병렬처리API

내부반복자

7. Stream을 이용한 반복문 처리

1. 반복문을 통한 일반적인 객체배열 처리

```
String objectArray[ ] = {"A","B","C","DX","E","F","G","H","I","J","K"};
for (String str : objectArray) {
        if (str.length( ) == 1) {
                System.out.println("I am "+str);
        }
}
```

2. Java8 객체 Stream 을 이용한 객체배열 처리

```
String objectArray[] = {"A","B","C","DX","E","F","G","H","I","J","K"};

// 1. 객체스트림으로 변경
Arrays.stream(objectArray)
        // 2. 길이가 1인것만 필터링
        .filter(a->a.length()==1)
        // 3. 반복을 통한 출력
        .forEach(a->System.out.println("I am "+a));
```


8.1. Stream을 추출하는 방법 1

1. 컬렉션에서 추출

```
Stream<String> strm = 컬렉션.stream();
```

2. 배열에서 추출

```
Stream<배열타입> strm = Arrays.stream(배열) 또는 stream(배열,from,to);
```

3. 숫자에서 추출

```
IntStream strm = IntStream.rangeClosed(1,1000);
```

4. 파일에서 추출

```
Stream<String> strm = Files.lines(파일경로, 문자셋);
```

5. 디렉토리에서 추출

```
Stream<Path> strm = Files.list(디렉토리경로);
```


8.2. Stream을 추출하는 방법 2

1. 배열값에서 추출

```
Stream<String> strm = Stream.of("스트링1","스트링2","스트링3");
```

2. 요소가 없는 stream 생성

```
Stream<String> strm = Stream.empty();
```

3. 무한 스트림 생성 (인자없는 함수형 인터페이스 Supplier<T> 객체)

```
Stream<String> new = Stream.generate(( ) -> "ANYTHING NEW");
```

4. 난수(Random Number) 스트림 생성

```
Stream<Double> rns = Stream.generate(Math::random);
```

5. 무한수열 스트림 생성 (seed 와 UnaryOperator<T> 객체를 받는다)

```
Stream<BigInteger> ints = Stream.iterator(BigInteger.Zero, n->n.add(BigInteger.ONE));
```


8.3. Sub Stream 추출과 Merge Stream

1. stream.**limit(n)** : n개의 스트림을 리턴

```
Stream<Double> rnds = Stream.generate(Math::random).limit(100); //난수 100개 생성
```

2. stream.**skip(x)** : 처음 x 번째까지 스트림을 제외

```
Stream<Double> rnds = Stream.generate(Math::random).skip(100); //처음 100개 제외
```

3. Stream.**concat(a, b)** : a 와 b 스트림을 병합

```
Stream<String> a = Stream.of("a","b","c","d","e");
Stream<String> b = Stream.of("e","f","g","h","i");

Stream.concat(a,b).forEach(System.out::println);
```


9. Stream 중간연산(intermediate operation)기법

- Stream 요소들을 최종 값이 도출되기 전에 필터링 하는 기법으로 **최종적으로 결과값을 도출하는 method가 호출되어야지만 적용**된다.

1. filter(람다식) - 필터조건을 만족하는 데이터만 추출

```
stream.filter( a -> a>10 );
```

2. distinct( ) - 중복제거

3. map??? 또는 flatMap???(람다식) -> 조건에 해당하는 스트림 데이터로 치환

```
stream.mapToInt( data -> 변경조건 );
```

4. boxed( ) - 값을 boxing 한다. int 를 Integer 객체로 double 을 Double 객체로 변환

```
intStream.boxed( );
```

5. sorted( ) - 값을 정렬한다. Comparable을 사용할 수 있다

```
intStream.sorted( ); // 숫자타입을 오름차순으로 정렬한다
```


10. Stream Looping(반복자)

- Stream 을 반복처리 하기위해서는 peek( ) 와 forEach( ) 두 가지 method를 사용할 수 있지만 peek는 중간처리를 목적으로 하기 때문에 결과처리를 하는 method가 호출되어야만 정상 적용이 된다.

1.peek( ) - 반복자이긴 하나 단일 호출로는 아무런 **결과값을 얻을 수 없다**

```
stream.peek( a -> System.out.println(a) ); // X 출력되지 않는다
                                           // Consumer 타입으로 리턴이 없다
// sum( ) 으로 결과값과 처리되는 중간값을 출력해주는 형태
// 따라서 중간 점검용 출력 형태로 사용된다
int total = stream.peek( System.out::println).sum();
```

2. forEach( ) - **결과처리 method**로써 단일호출로도 **결과값을 얻을 수 있다**

```
// 람다식에 의해 반복문을 돌면서 값을 출력해준다
stream.forEach( a -> System.out.println("반복 :"+a ) );
```


11. Stream 결과연산(terminal operation)기법

1. Matching - 특정 조건에 부합하는지 여부를 boolean 값으로 리턴

```
// 전체 비교
stream.allMatch(a -> a > 3 ); // stream 요소 전부가 3보다 크면 true
// 일부 비교
stream.anyMatch( a -> a > 3); // stream 요소 중 하나라도 3보다 크면 true
// 전체 비교 부정
stream.noneMatch( a -> a > 3); // stream 요소 전부가 3보다 크지 않으면 true
```

2. Reduction method - sum( ), count( ), average( ), max( ), min( )

```
int total = stream.sum( );    // 합계
int count = stream.count( ); // Stream 요소의 개수

int max = stream.max( ).getAsInt( ); // 최대값
int min = stream.min( ).getAsInt( ); // 최소값
int avg = stream.average( ).getAsDouble( ); // 평균값
```

리턴값이 Optional 이기 때문에 getAs로 기본형으로 변환해야 한다

12. Optional

- Optional은 값이 있거나 없을을 나타낼 수 있는 객체 타입이다
  기존 자바에서는 null 포인터로 값이 있고 없음을 나타내었지만 Java8 에서는
  Optional 을 통해 값의 있고 없음을 판단 할 수 있다

1. Optional 생성

```
Optional<String> opt1 = Optional.empty(); // 빈객체 생성
Optional<String> opt2 = Optional.of(str);     // str이 null일 경우 NullPointerException 발생
Optional<String> opt3 = Optional.ofNullable(str);  // Null을 허용하는 optional
```

2. Optional 연산

```
// 값이 존재하는지 체크
if(opt3.isPresent()){
        System.out.println(opt3.get());
}
// 값이 존재하는지 체크하는 내부메서드
opt3.ifPresent(a -> System.out.println(a));  //또는 opt3.ifPresent(System.out::println);

// 값이 없을경우 default 값 설정
String result_default = opt3.orElse("NONE"); // 빈객체일경우 문자열 "NONE"을 리턴
```


13. Grouping

- groupingBy : 성질이 같은 것들끼리 그룹을 묶어주는 분류함수(classifier function)

1. 로케일을 국가별로 묶어주는 예제

```
Stream<Locale> locales = Stream.of(Locale.getAvailableLocales());

Map<String, List<Locale>> localesOfCountries =
            locales.collect(Collectors.groupingBy(Locale::getCountry));
```

2. 특정 로케일(IE=아일랜드) 출력하기

```
localesOfCountries.get("IE").forEach(System.out::println);
```

// ga_IE, en_IE 두 개가 출력된다

* 로케일이란?
언어코드와 국가코드를 포함하는 것으로 위의 예제에서 국가코드인 IE를 기준으로 로케일을 출력하면 아일랜드(IE)에서 사용하는 두 가지언어 코드인 ga(게일어)와 en(영어) 이 국가코드와 함께 출력된다

