모던 JAVA

함수형

- 익명 클래스의 번거로움을 람다로 간편하게, 메서드 참조로 재사용
- 코드 블록을 주입(동작 파라미터화)하고 조합할 수 있게 됨
- 스트림의 기반, 병렬처리와 조화

람다, 메서드 참조

- 람다 : 익명 함수, 익명 클래스를 대체
- 함수형 인터페이스(이름 있는 람다) : 하나의 추상 메서드를 가진 인터페이스
- 메서드 참조 : 메서드나 생성자를 참조하기(::)

→ EX) 문자열 리스트를 길이에 따라 정렬

```
//옛날 버전
Collections.sort(words, new Comparator<String>(){
    public int compare(String o1, String o2){
        return Integer.compare(o1.length(), o2.length());
    }
};

//java 8버전
Collections.sort(words, (o1, o2) -> Integer.compare(o1.length(), o2.length()));

//메서드 참조
Collections.sort(words, Comparator.comparingInt(String::length));

//java 9버전
words.sort(Comparator.comparingInt(String::length));
```

스트림

- 컬렉션 + 함수형, 데이터 처리 연산을 지원하도록 소스에서 추출된 연속된 요소
  - 외부 순환 VS 내부 순환
  - SQL 처럼 선언형 스타일로 데이터를 처리
  - 쉽게 병렬처리 적용 : parallelStream 메서드

- 주요 패키지
  - `java.util.stream`
- 주요 클래스
  - `BaseStream` `Stream`
- 주요 메서드
  - `map()` `filter()` `reduce()` `min()`
  - `C.stream()` `C.parallelStream()`

- 주요 개념
  - 중간 연산과 최종 연산
    - 중간 연산 : 스트림을 반환, 여러 연산을 조합할 수 있음
    - 최종 연산 : 스트림을 모두 소비하고 닫음

- 스트림은 1회용 → 최종 연산 이후 사용 불가

⇒ EX)

```
//직원 리스트 -> 부서별 직원 리스트
Map<Department, List<Employee>> byDept = employees.stream().collect(Collectors.groupingBy(Employee::getDepartment));

//직원 리스트 -> 부서별 급여 합계
Map<Department, Integer> totalByDept = employees.stream().collect(Colloetors.groupingBy(Employee::getDepartment, Collectors.summingInt

//좋은 직원 안좋은 직원 나누기
Map<Boolean, List<Employee>> byGooe = employees.stream().collect(Colloectores.partitioningBy(Employee::isGood));
```

병렬 / 동시성

- 많이 사용되는 패턴들을 언어 차원에서 API로 지원
- 고수준, 추상화, Thread Safety, 비동기 지원

- 주요 패키지
  - `java.util.concurrent`
- 주요 클래스
  - `Executor(s)` `ExecutorService`
  - `xxThreadPool` `ForkJoinPool`
  - `Future` `CompletableFuture`
  - `Runnable` `Callable`

Executor / Service / Etc

- Thread를 직접 생성, 관리하지 않고 ExecutorService 에서 스레드 받기
- 작업(Runnable, Callable)을 Executor 서비스에 요청하고 결과 받기
- 작업 스케줄링 기능 ⇒ ScheduledExecutorServcie
- Concurrent Collection
  - Thread Safe List / Map 제공
- Atomic Variable
  - 변수 자체가 원자성을 보장
- Lock 객체
  - 동기화 패턴에 따라 사용할 수 있는 유틸리티

비동기 지원

- 동기 VS 비동기 & 블록 VS 넌블록
- Future : 비동기 연산 지원, 완료 확인/ 대기/ 결과 조회/ 취소
- CompletableFuture : Future 작업 연결, 순서 정의 등