250902_멋사_써머리

***로 표기된것은 다음에 나와요~

1.정렬

아래 종류는 뒤에나옴
버블 정렬인접한 두 원소를 반복적으로 비교해 정렬하는 방식으로, 비효율적이지만 가장 단순하다.
삽입 정렬원소를 하나씩 뽑아 이미 정렬된 배열의 올바른 위치에 삽입하는 방식으로, 데이터가 거의 정렬된 상태일 때 빠르다.
합병 정렬배열을 반으로 계속 나누어 정렬하고 합치는 방식으로, 안정적이며 데이터의 크기와 상관없이 일정한 성능을 보인다.
퀵 정렬'피벗'이라는 기준 원소를 정하고, 그보다 작거나 큰 원소들을 나눠 재귀적으로 정렬하는 방식으로, 가장 빠르다고 알려져 있다.

2. Collection vs Collections

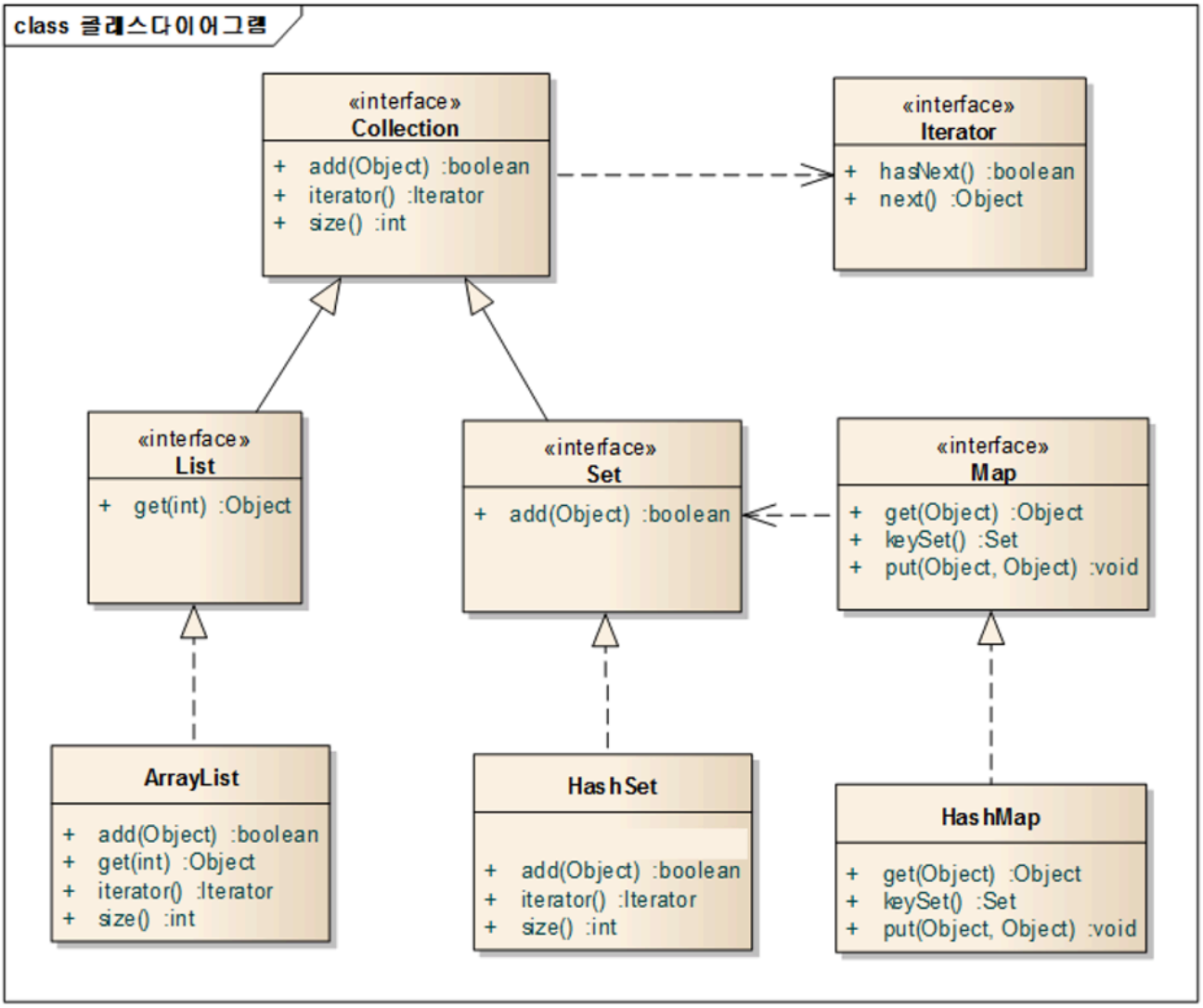
- Collection은 객체들이 무엇을 할 지에 따라 적절한 자료구조를 선택
- Collections는 컬렉션 객체들을 편하게 다룰 수 있기 위해 존재

	collection	**collections**
종류	인터페이스	클래스
메소드	List, Set, Queue	sort(), reverse(), shuffle()
역할	데이터구조정의	컬렉션 조작을 위한 메서드 제공
객체생성	X	X

- 비슷한느낌으로 array는 배열 인데 arrays는 배열의 도구모음으로 사용*

mysql 마이시퀄 @애노테이션 (개발자들 그냥 이렇게 부름)
매개변수에 어떤타입이 들어가나요? 라는 말의 의미

- 매서드/생성자 정의시 매개변수가 들어가는데 이때 들어가는 타입이 어떤건지 물어보는것
  (int, string 등등등등등 더 쉽게 한글인지 숫자인지 숫자면 어떤 숫자인지 int, long 등)



자료구조	대표 클래스 / 인터페이스	특징	삽입/삭제	탐색/접근	사용 용도
배열(Array)	`int[]`, `String[]`	고정 길이, 연속 메모리, 인덱스로 접근 가능	O(n) (중간 삽입/삭제)	O(1) (인덱스 접근)	순차적 데이터 저장, 빠른 조회
ArrayList	`ArrayList<E>`	동적 배열, 순서 유지, 중복 허용	O(n) (중간 삽입/삭제)	O(1) (인덱스 접근)	조회 중심, 순차적 데이터 저장
LinkedList	`LinkedList<E>`	양방향 링크드 리스트, 순서 유지	O(1) (앞/뒤), O(n) (중간)	O(n)	삽입/삭제 잦은 경우, 큐/스택 구현
Stack	`Stack<E>` 또는 `Deque<E>`	LIFO, push/pop	O(1)	O(n)	되돌리기, 후입선출 구조

자료구조	대표 클래스 / 인터페이스	특징	삽입/삭제	탐색/접근	사용 용도
Queue	`Queue<E>`, `LinkedList<E>`	FIFO, offer/poll	O(1)	O(n)	순차적 처리, 작업 대기열
PriorityQueue	`PriorityQueue<E>`	힙 기반, 우선순위 정렬	O(log n)	O(n)	우선순위 큐, 작업 스케줄링
HashSet	`HashSet<E>`	중복 X, 순서 X, 해시 기반	O(1)	O(1)	유일한 값 저장, 집합 연산
LinkedHashSet	`LinkedHashSet<E>`	입력 순서 유지	O(1)	O(1)	순서 있는 유일 값 저장
TreeSet	`TreeSet<E>`	정렬된 집합, 이진 탐색 트리 기반	O(log n)	O(log n)	정렬 필요 집합, 범위 검색
HashMap	`HashMap<K,V>`	키-값 저장, 순서 X, 해시 기반	O(1)	O(1)	검색/조회 중심, 캐시
LinkedHashMap	`LinkedHashMap<K,V>`	입력 순서 유지	O(1)	O(1)	순서 있는 키-값 저장
TreeMap	`TreeMap<K,V>`	정렬된 키-값, 이진 탐색 트리 기반	O(log n)	O(log n)	정렬된 조회, 범위 검색
Hashtable	`Hashtable<K,V>`	동기화된 해시맵, 구식	O(1)	O(1)	스레드 안전 필요 시 (현실적으론 잘 안 씀)

3.SORT

- 정렬 기준은 개발자가 정한다!
  3.1 Comparable vs Comparator
- 정렬 기준을 세울 때 상황에 따라 적절한 것을 사용

🆚 한눈에 비교

구분	Comparable	Comparator
패키지	`java.lang`	`java.util`
비교 기준 위치	클래스 내부	클래스 외부 (별도 객체)
메서드	`compareTo(T other)`	`compare(T o1, T o2)`
정렬 기준 변경	클래스 수정 필요	새로운 Comparator 작성하면 됨
사용 예시	`Collections.sort(list)`	`Collections.sort(list, comp)`

💡 정리

- **Comparable** → 그 객체가 "자기 자신이 어떻게 정렬될지" 기본 기준을 제공.
- **Comparator** → 외부에서 "정렬 기준을 다양하게" 제공할 수 있게 해줌.

```
public class PersonComparator implements Comparator<Person> {
    @Override
    public int compare(Person o1, Person o2) {
        return o1.getAge() - o2.getAge();
    }
}
```

```
Collections.sort(persons, new Comparator<Person>() {
            @Override
            public int compare(Person o1, Person o2) {
                return o1.getAge() - o2.getAge();
            }
        });
```

*** 일급 객체(First-class object)란?(안중요함)

- 값처럼 자유롭게 다룬다.
- 즉, **변수에 담을 수 있고, 매개변수로 넘길 수 있고, 함수의 반환값으로도 쓸 수** 있어야 합니다.

1. **변수나 데이터 구조 안에 저장할 수 있어야 한다.**
2. **함수(메서드)의 인자로 전달할 수 있어야 한다.**
3. **함수(메서드)의 결과로 반환할 수 있어야 한다.**

🔹 비교 예시

종류	변수에 담기	인자로 넘기기	반환값으로 쓰기
일급 객체	✅	✅	✅
이급 객체	❌	✅	❌
삼급 객체(Third-class)	❌	❌	✅ (매우 드묾)

※ 삼급 객체는 거의 학술적인 개념이고 실무에서는 거의 쓰이지 않아요.

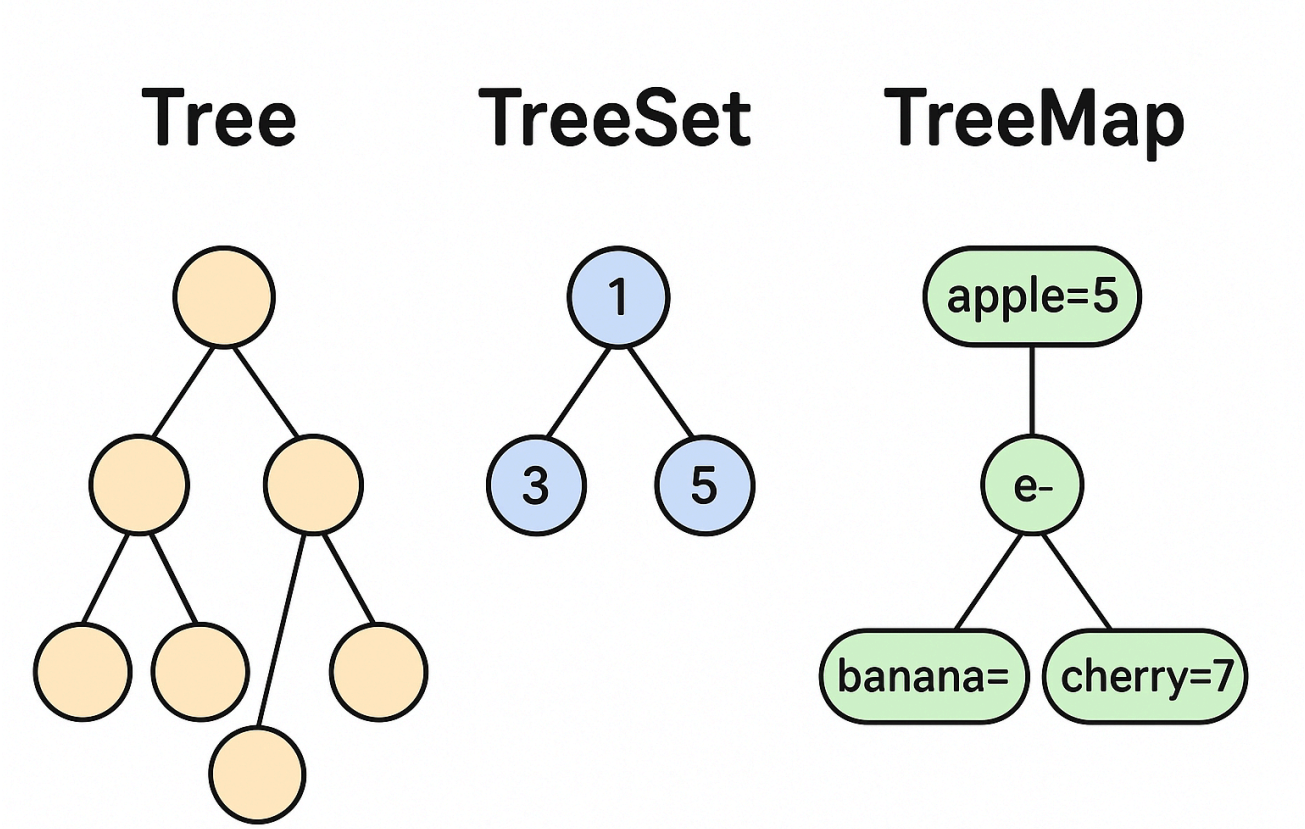
*4. 트리구조**(중요한데 뒤에 또나옴)

자료구조	중복 허용	자동 정렬	사용 목적
Tree (concept)	노드에 따라 달라짐	직접 구현 필요	트리 구조 일반
TreeSet	❌	✅	정렬된 Set, 중복 없는 집합
TreeMap	키 ❌, 값 ✅	✅	키 기준 정렬된 Map, 검색/삽입/삭제 효율적

💡 **Tip:**

- 단순히 정렬된 집합이 필요하면 → `TreeSet`
- 키-값 쌍이 필요하고 키 기준으로 정렬해야 하면 → `TreeMap`

- 직접 트리 자료구조를 만들고 싶으면 → Node 기반 Tree 구현



5. 제네릭<> :과 Object 박스 차이점

<1>제네릭

- 정해진 타입만 사용하도록 강제할 수 있습니다.
- 타입을 강제함으로써 컴파일할 때 잘못된 타입의 값이 저장되는 것을 막을 수 있습니다.
- 형변환(casting)이 불필요합니다.
- 컴파일 타임에 타입 안전성을 보장합니다.

<2>ObjectBox**

- 어떤 Object든 저장할 수 있고, 어떤 Object를 꺼낼 수 있습니다.
- 꺼내서 사용할 때는 원래 타입으로 변환시키는 번거로운 과정이 필요합니다.
- 잘못된 타입으로 캐스팅할 경우 런타임 오류가 발생할 수 있습니다.

6. 칼렌다

- 칼렌다는 왜 추상클래스로 만들었나?
  다형성(공통기능과 다양한 구현이 요구됨)
- 강사님 생신 11/24
- Calendar.getInstance(); 인스턴스화 안된다

- 그레고리안 캘린더
- 요새 유행은 아래 메서드 사용!
  `// LocalDateTime - 사람이 읽는 날짜, 시간 (타임존 없음)

// ZonedDateTime - 사람이 읽는 날짜.시간 + 타임존 (지역반영)

// Instant - 컴퓨터가 읽을 목적. 항상 UTC기준 (서버/DB/로그 기록등에 자주쓰임)

**예외처리 예제
NullPointException
NumberFormatException
IllegalArgumentException
IndexOutofBound

에러:수습할 수 없다. //산사태
예외:수습가능 //정상적인 실행 도중 갈 길을 잃은것

남혜린

9월 2일 회고록
오늘은 오전에 Collection과 Comparable에 대해 배웠다. 다행히 큰 무리 없이 수업을 따라갈 수 있어서 뿌듯했다. 오후에는 예외처리에 대해 설명을 들었는데, 내일은 예제를 다룬다고 하셔서 오늘 꼭 복습을 해둬야겠다고 다짐했다. 미리 준비를 해두면 내일 수업도 무리 없이 소화할 수 있을 것 같다. 프로젝트를 진행할 때 체력이 많이 떨어지는 걸 느껴서 조금 걱정이 된다. 그래서 수업 후 운동을 하며 체력도 관리해야겠다. 공부도 중요하지만 꾸준히 달리려면 건강이 뒷받침되어야 하니까. 이번 주에 목표해둔 분량까지 문제없이 잘 달려가는 게 최우선이다. 끝까지 집중해서 달리고, 체력 관리도 함께 챙기면서 흔들림 없이 나아가야겠다

윤수정

2025/09/02
오늘은 역대급으로 힘든 하루였다. 몸이 무겁고 정신적으로도 지친 하루를 보내면서, 무엇보다 건강을 잘 챙겨야 한다는 생각이 크게 다가왔다. 공부와 개발을 이어가는 것도 결국 몸과 마음이 버텨주어야 가능한 일이기에, 앞으로는 균형 잡힌 생활과 체력 관리에 더 신경 써야겠다고 다짐했다. 작은 습관 하나라도 놓치지 않고 지켜내는 것이 장기적으로 큰 힘이 될 것이라 생각한다.

그리고 역시 단순히 강의를 듣고 필기하는 수준으로는 부족하다는 것을 알았다. 컬렉션과 자주 쓰이는 API들을 실제로 코드로 작성해보고 반복해서 손에 익히는 과정이 필요하다. 이 과정을 통해 단순한 지식이 아니라 몸에 새겨진 실력으로 바꿔가야 한다. 오늘 느낀 부족함과 피로가 결국 나를 더 단단하게 만드는 동력이 되기를 바란다. 화이팅!오늘은 역대급으로 힘든 하루였다. 몸이 무겁고 정신적으로도 지친 하루를 보내면서, 무엇보다 건강을 잘 챙겨야 한다는 생각이 크게 다가왔다. 공부와 개발을 이어가는 것도 결국 몸과 마음이 버텨주어야 가능한 일이기에, 앞으로는 균형 잡힌 생활과 체력 관리에 더 신경 써야겠

다고 다짐했다. 작은 습관 하나라도 놓치지 않고 지켜내는 것이 장기적으로 큰 힘이 될 것이라 생각한다.
아자!

백종현
오늘의 회고 2025-09-02

오늘 정렬에 대한 부분과 제네릭 타입에 대한 부분을 배웠습니다.
두 부분 다 매우 중요한 부분이라 이번 강의 내용을 잘 복기 해야 할것 같습니다
팀원 분들이 모두들 잘 이해 하신것 같아 다행이였고 저도 뒤처지지 않게 열심히 공부해야겠단 자극이 들었습니다!

오늘의 키워드

- 정렬
- Collections
- Comparable
- Comparator
- Generic
- 일급 객체

진솔빈

2025 . 09 . 02 오늘의 회고
오늘은 조금은 생소하지만, 들어는 봤던 자료구조의 일부인 STACK 과 QUEUE구조를 배우면서 LIFO, FIFO를 배웠고, Tree 구조 일부도 배웠다. 이제, JAVA도 곧 파일입출력을 배울 것이고, 거의 다 배워간다. 그 다음인 DB도 배울 예정이지만, 아직은 시간이 조금만 더 있었다면 좋겠다는 소망이 있다. 아쉽긴 하지만, 얼른 부족한 부분을 메꿔 나가야겠다. 사실, 어느 정도 익숙해지긴 했지만, 아직 구멍들이 곳곳에 보여 자바의 모든 과정이 끝나기 전에, 늦더라도 이번 주말까지 껴서 이번주 안에는 꼭 자바를 익숙하고 원활하게 다루기 위한 노력이 필요한거같다.

최형규
9월 2일 오늘의 회고

미니플젝 때 ai 없이 에러가 나더라도 일일히 쳐보고 틈틈히 내가 이걸 어떻게 쓰는지 머리를 꽁꽁 싸맨 결과!
정말 오랜만에 간만에? 수업들으면서 실행시키는데도 그럭저럭 잘따라가는 나를 발견했다.
이게 미니플젝 효과인것인가! 체력소모는 최상이었지만, 무언가 자신감도 붙고 좋았다.
근데 아직 근데 아직 근데... 아직 정상컨디션은 아닌듯 하다.
아무튼! 다들 좀 피곤한 기색이 있었지만, 학습열의가 아직까지 불타오르고 있는 모습을 보니
나도 거기에 휩쓸려서 재밌게 한거같다.
키워드 아직 조금 어려운 것도 있지만 하나하나 직접 코드도 쓰고 내껄로 만들자

오늘의 키워드

- 정렬(종류가 많다.)
- Colloection(s)
- Comparable vs Comparator
- 제네릭<>
- 트리구조(차차)
- Date
- Calendar
- Time
- 예외처리
  ㄴTry, Catch