

객체지향 프로그래밍 및 실습

12주차. Generic Collections

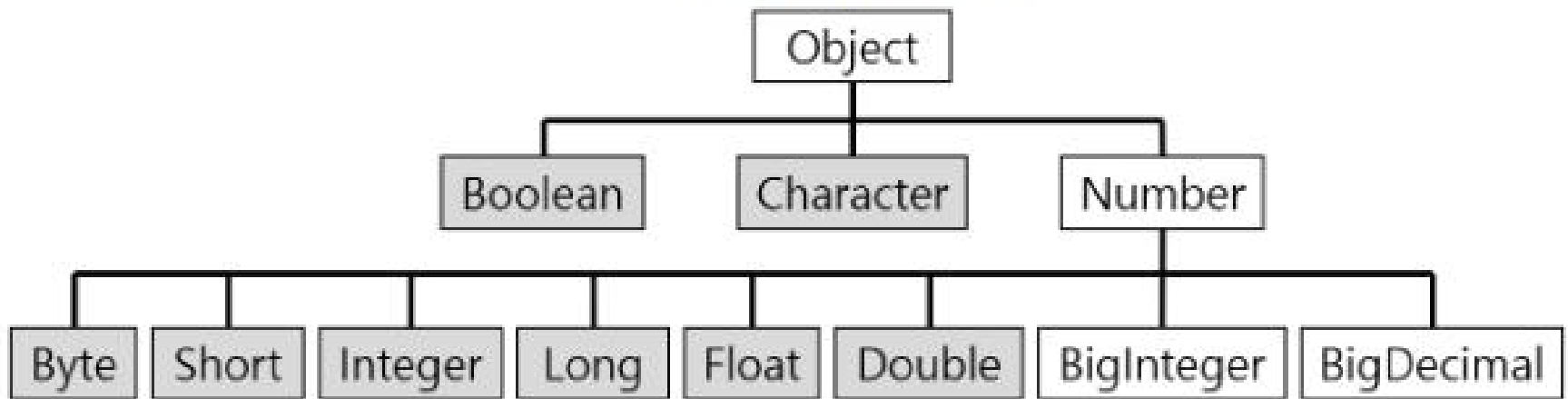
1. Type Wrapping

■ What is Type Wrapping

- 자바에서의 타입은 원시형과 참조형으로 나뉜다.
 - 원시형: int, double, boolean, ... 변수가 담고 있는 데이터가 그 자체이다. - int는 정수를 담음
 - 참조형: new 연산자로 생성된 모든 것 - List<?>는 객체의 레퍼런스를 담고 있다.
- List<A>에서의 A는 참조형만 사용할 수 있다.
 - List<Student>, List<String> 등
 - 그렇다면 int, double 등 원시형을 담을 수는 없을까?
 - 또는 원시형을 참조형으로 사용할 수는 없을까?
- 이럴 때 사용하는 것이 Type Wrapper class

1. Type Wrapping

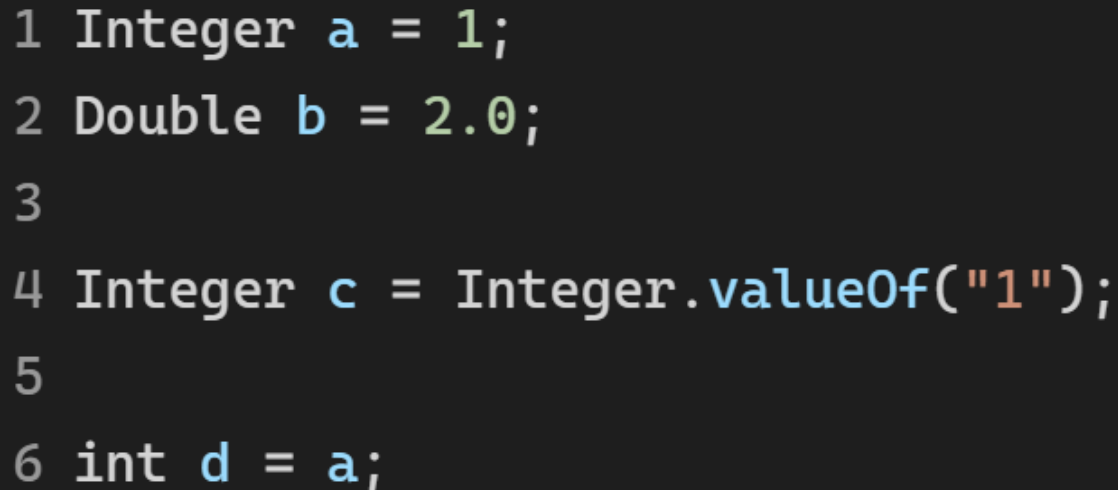
- Type Wrapper class
 - Type wrapper class의 구조



- BigInteger: Immutable, arbitrary-precision integers.
- BigDecimal: Immutable, arbitrary-precision signed decimal numbers.

1. Type Wrapping

- Wrapper class usage
 - 원시형으로 초기화 할 수 있다.
 - Integer.valueOf() 등으로 사용할 수도 있다.




```
1 Integer a = 1;  
2 Double b = 2.0;  
3  
4 Integer c = Integer.valueOf("1");  
5  
6 int d = a;
```

1. Type Wrapping

- Wrapper class usage

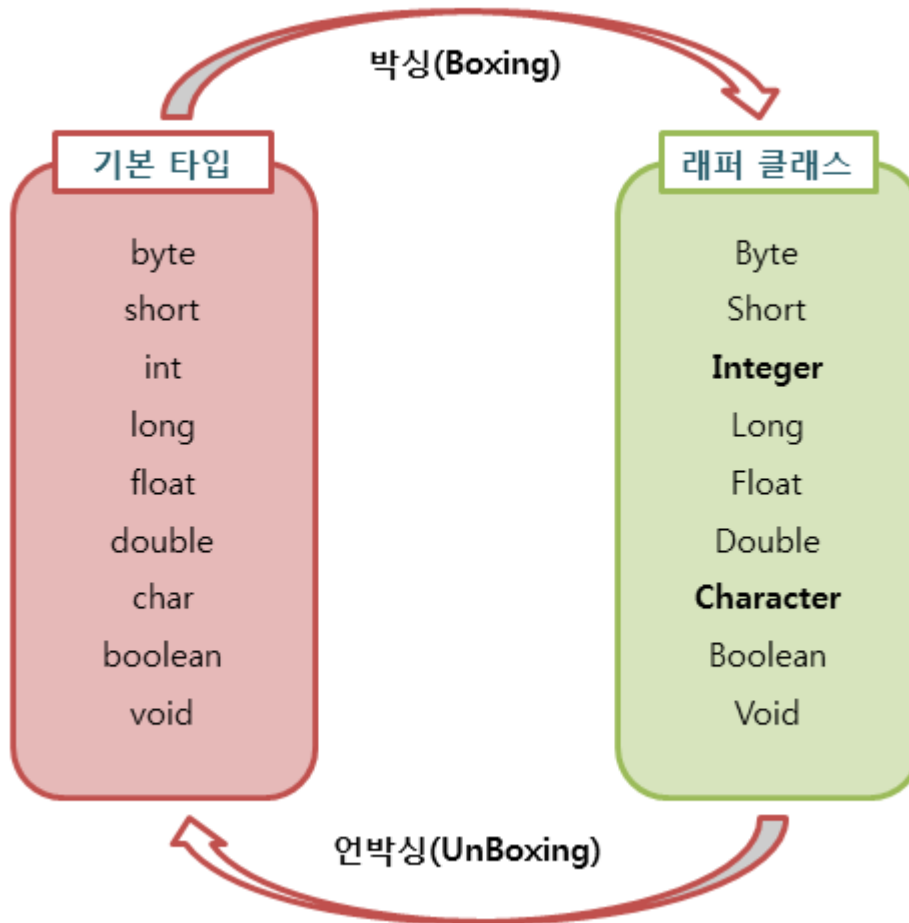
- 다양한 utility를 제공하기도 한다.
- 자세한 내용은 직접 이클립스의 코드 힌트를 사용해보자



```
1 int max = Integer.max(d, a);  
2 Double.isFinite(Double.POSITIVE_INFINITY);  
3 Boolean.logicalAnd(false, true);
```

1. Type Wrapping

■ Auto Boxing



```
1 Integer boxing = 5; // auto boxing
2 int unboxing = boxing; // auto unboxing
```

1. Type Wrapping

■ .equals

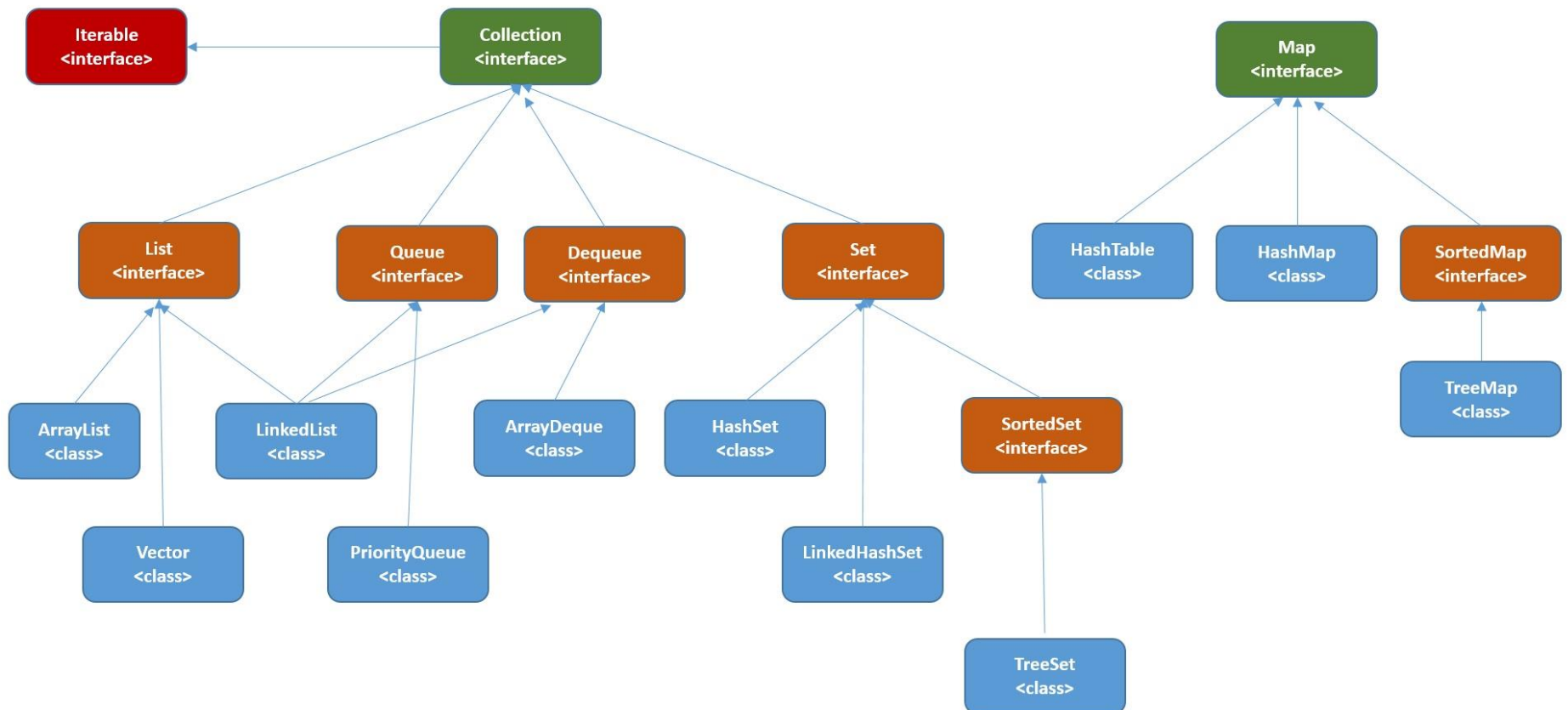
- Wrapper 클래스가 같은 수를 담고 있더라도 기본적으로는 참조형이기 때문에 내부의 value로 == 연산을 사용할 수가 없다.
- 아래에서는 레퍼런스(주소 값과 비슷) 끼리 비교 연산을 하는 것에 불과
- 따라서 6번 라인은 false를 출력하며 x1.equals(x2)를 사용하는 것이 적절
- String.equals와 같은 경우

```
1 // Wrapper class에 new 연산자를 사용하는 것은 deprecated 되었기 때문에 권장하지 않음
2 // 다만 실습 예제를 위해서 사용
3 Integer x1 = new Integer(10);
4 Integer x2 = new Integer(10);
5
6 System.out.println(x1 == x2);
```

2. Collections Framework

■ Collections Framework의 구조

Collection Framework Hierarchy



2. Collections Framework

- Collections Framework

- 컬렉션 프레임워크는 다양한 컬렉션을 표현하고 조작하기 위한 통합 아키텍처
- 모든 컬렉션 프레임워크는 3가지를 포함한다.
 - Interface: 컬렉션을 나타내는 추상 abstract data type (ex. Map, List, Set...)
 - Implementations: Collection Interface에 대한 concrete 구현체 (ex. HashMap, ArrayList, ...)
 - Algorithms: 컬렉션에 대해서 정렬이나 서칭 등의 연산을 수행하기 위한 메서드

2. Collections Framework

- Iterator

- 컬렉션을 순회할 수 있도록 해주는 객체

```
1 List<Integer> list = new ArrayList<>();
2 for (int i = 0; i < 10; i++) {
3     list.add(i);
4 }
5
6 Iterator<Integer> iter = list.iterator();
7
8 while (iter.hasNext()) {
9     System.out.println(iter.next());
10 }
```

3. List

■ List

- Ordered Collections – sequence 라고 부른다.
- 대부분의 상황에서 ArrayList가 최고의 성능을 보이며, 특별한 상황에서는 LinkedList가 더 나은 성능을 보이기도 한다.
- 기능
 - Positional access
 - Search
 - Iteration
 - Range-View

3. List

■ List

- List는 순서가 있는 컬렉션이기 때문에 정렬이 가능하다.
- 주로 Comparable을 구현하거나 익명클래스, 람다식을 사용한다.

```
1 List<Student> list = new ArrayList<>();
2
3 for (int i = 0; i < 10; i++) {
4     if (i % 2 == 0) {
5         list.add(new Student(i*2));
6     } else {
7         list.add(new Student(i));
8     }
9 }
10
11 list.forEach(s → System.out.println(s.getAge()));
12
13 list.sort((o1, o2) → {
14     if (o1.getAge() > o2.getAge()) {
15         return 1;
16     } else if (o1.getAge() == o2.getAge()) {
17         return 0;
18     } else {
19         return -1;
20     }
21 });
22
23 list.forEach(s → System.out.println(s.getAge()));
```

3. List

■ 실습 문제 1

- StockAccount 객체를 리스트에 담고 이를 정렬해보자
- StockAccount - 주식 계좌 클래스
 - 필드
 - int amt; // 보유 주식 수
 - int price; // 현재가
 - String name; // 종목 명
 - 정렬 기준
 - amt * price -> 해당 종목에 대한 총 평가 금액
 - 총 평가 금액을 기준으로 정렬해보자
- 아래의 객체를 리스트에 순서대로 담아서 정렬한다.
 - amt: 5, price: 1550, name: 현대건설
 - amt: 50, price: 77500, name: 삼성전자
 - amt: 87, price: 3650, name: 한국콜마
 - amt: 600, price: 50, name: 비트코인
 - amt: 900, price: 200, name: 민영정보통신
 - Amt: 50, price: 700, name: 바나스

4. Map

■ Map

- Key, value 쌍이 있어, key로 접근하는 컬렉션

```
1 Map<String, String> studentMap = new HashMap<>();  
2  
3 studentMap.put("201921166", "정의철");  
4 studentMap.put("202213546", "김땡땡");  
5 studentMap.put("201851354", "박땡땡");  
6  
7 System.out.println(studentMap.get("202213546"));
```

4. Map

■ 실습 문제 2

- word.txt를 읽어서, 각 단어의 빈도수를 측정해보자.

```
1 Scanner scan = new Scanner(new File(System.getProperty("user.dir"), "word.txt"));
2 String text = scan.nextLine();
3 text = text.toLowerCase();
4
5 String[] wordList = text.split(" ");
6
7 Map<String, Integer> wordFreq = new HashMap<>();
8 // ...
9
10 scan.close();
```

4. Map

■ Map의 순회

- Map.entrySet()을 이용하여 순회할 수 있다.

```
1 for (Map.Entry<String, Integer> freqSet : wordFreq.entrySet()) {  
2     System.out.printf("%s = %d\n", freqSet.getKey(), freqSet.getValue());  
3 }
```

- 람다식을 이용한 순회

```
1 wordFreq.entrySet().forEach(e → System.out.printf("%s = %d\n", e.getKey(), e.getValue()));
```


5. Set

■ Set

- Set은 Unordered, unique element를 담기 위한 컬렉션이다.
- 우리가 알고 있는 '집합'의 개념과 같다.

■ HashSet

- 객체의 hashCode() method를 통해 객체가 동일한지 판단
- String 타입인 경우 문자열이 같으면
hashCode()와 equals() return 값이 항상 일치하도록 method를 override함

```
import java.util.HashSet;
import java.util.Iterator;

public class HashSetExample {
    public static void main(String[] args) {
        HashSet<String> set = new HashSet<String>();

        set.add("김성준");
        set.add("김국디");
        set.add("이아주");
        set.add("김성준");

        System.out.println("set size: " + set.size());

        Iterator<String> iter = set.iterator();
        while(iter.hasNext()) {
            System.out.println(iter.next());
        }
    }
}
```

```
<terminated> HashSetExample
set size: 3
김국디
김성준
이아주
```

5. Set

- 실습 문제 3

- setTest.txt 파일을 읽어서, distinct word만을 뽑아서 출력해보자

6. Queue

■ Queue

- FIFO 구조를 가지는 컬렉션

```
1 Queue<String> queue = new LinkedList<>();
2 queue.add("Student1");
3 queue.add("Student2");
4 queue.add("Student3");
5
6 System.out.println(queue.poll());
7 System.out.println(queue.poll());
8 System.out.println(queue.poll());
```

- 출력은 집어넣은 순서대로

7. Deque

■ Deque

- Deck 라고도 부르는 구조로써, double-ended queue 컬렉션

```
1 Deque<String> deck = new LinkedList<>();  
2  
3 deck.add("Student1");  
4 deck.addFirst("Student2");  
5 deck.addLast("Student3");  
6  
7 System.out.println(deck.pollFirst());  
8 System.out.println(deck.pollLast());
```

7. Deque

■ 실습 문제 4

- 컬렉션을 조합해보자.
- `Map<String_단과대, Map<String_학과, List<String_학생>>>`
- 정보통신대학 안에 (소프트웨어학과, 국방디지털융합학과, 등...)
- 국방디지털융합학과 안에 학생 리스트(김땡땡, 박땡땡, ...)