

# Lab 13

# Q&A

<https://padlet.com/dcslabcp/0607-dzhnbgt9gp1wjvvn>

# Exercise 5 주요 질문들

Q : deposit(입금) 메서드는 소지한 현금을 계좌에 입금하는 것인가요, 아니면 단순히 잔고를 늘리는 것인가요?

Q : 행6에서 입금을 실행할 때 자신의 현금에서 빼가는 것인가요? 실행4를 보면 잔고에서 뺀 돈이 현금에 추가되는데 실행6은 현금과 잔고의 관계가 나오지 않아 질문드립니다

Q: ex5 문서에서 NegativeBalanceException 부분에서 "잘못된 금액"이라는 message가 포함되어 있는데, 올려주신 실행 예시들에서 어떤 경우에 "잘못된 금액" exception이 되는지 찾기 어려워서 질문 드립니다.

A : 빠뜨린 case들을 찾아주셔서 감사합니다.

# Exercise 5 주요 질문들

Q : 과제5 안내 pdf에 transfer method를 user class에서 구현하라고 되어있는데, Bank class에서 해당 메소드를 구현하여도 괜찮을까요? 두 user object에 대한 정보가 필요한 operation이라 bank class에서 구현하는 것이 더 직관적이고 효율적으로 코드를 작성할 수 있을 것 같아 질문드립니다.

Q: Optional이나 Collection의 Stream 같은 기능을 사용하면 표현력이 좋아질 것 같은 부분들이 있는데 이런 기능들을 사용해도 괜찮을까요?

과제 채점에 있어서 자바 버전에 대한 제약이 있을까요?

A : 더 좋은 코드는 언제나 환영합니다.

채점이 우려되신다면, 본인이 사용한 java 버전을 코드 상단에 주석으로 명시해주시면 도움이 됩니다.

# Exercise 5 주요 질문들

Q : data.txt에서 내용을 읽어 온 후 프로그램 종료 전 업데이트 된 상황을 data.txt에 반영해야 하는지 궁금합니다.

Q: 인출액/입금액/송금액/물건 가격이 0 이상인지, 0 초과인지 궁금합니다.  
자기 자신에게 송금할 수 있는지 궁금합니다.

A : 과제가 다 담지 못한 부분을 잘 찾아 주셨네요.

Exercise로 낸 과제는 간단한 연습문제라고 생각해주시고,  
더 크고 복잡한 프로그램을 만드실 때에는 최대한 다양한 경우를 고려하는 게 좋습니다.

# Quiz 1

- 다음 두 개의 클래스를 어떻게 Box라는 클래스로 통합할 수 있을까요?
- quiz package의 클래스들을 이용해 Box 클래스를 완성해보자

**public class Box1**

**public class Box2**

// String 객체만 저장할 수 있는 Box 클래스

Box1
- data : String
+ set(String) : void + get() : String

// Integer 객체만 저장할 수 있는 Box 클래스

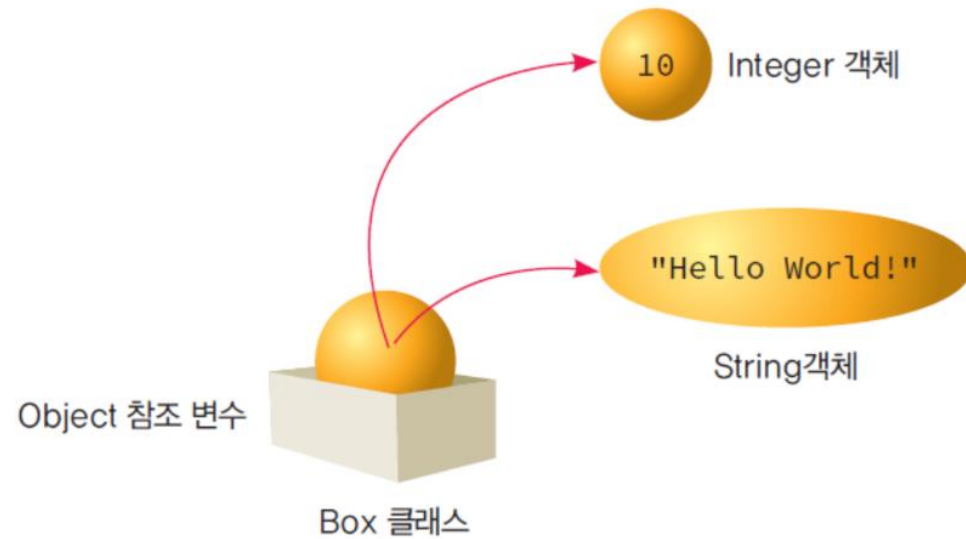
Box2
- data : Integer
+ set(Integer) : void + get() : Integer

# Quiz 1

```
public class Box {  
    2 usages  
    private Object data;  
    public void set(Object data) { this.data = data; }  
    no usages  
    public Object get() { return data; }  
}
```

# Object 상향 형변환

- Object 클래스를 이용한 상향 형변환
- Object 클래스를 이용하면 어떤 객체이든지 참조 가능





# Quiz 2

- Box 클래스에 아래 main 함수를 추가해서 실행해서 결과를 확인하고 padlet에 답을 남겨보자.

```
public static void main(String[] args) {  
    Box b = new Box();  
    b.set("Hello World!");           // 1 문자열 객체를 저장  
    String s = (String)b.get();      // 2 Object 타입을 String 타입으로 형변환  
  
    b.set(new Integer(10));          // 3 정수 객체를 저장  
    Integer i = (Integer)b.get( );  // 4 Object 타입을 Integer 타입으로 형변환  
  
    s = (String)b.get();  
}
```

- 혹시 에러가 나시는 분은 에러메시지와 함께 게시판에 글을 남겨주시기 바랍니다.

# Quiz 2

```
Exception in thread "main" java.lang.ClassCastException: Create breakpoint : class java.lang.Integer cannot be cast to class java.lang.String  
at Box.main(Box.java:15)
```

- 일반 예외와 실행 예외 중 어떤 것일까요?      → 실행 예외(Runtime Exception)

# Generic Class

- 제너릭 클래스에서는 타입을 변수로 표시하고, 이를 타입 매개변수(type parameter)라 함
- 타입 매개변수는 객체 생성 시 프로그래머에 의하여 결정됨
- 하나의 코드를 다양한 타입의 객체에 재사용 가능
- 클래스, 인터페이스, 메서드를 정의할 때 타입을 변수로 사용

```
class 클래스이름<타입매개변수> {
```

```
    필드;  
    메서드;
```

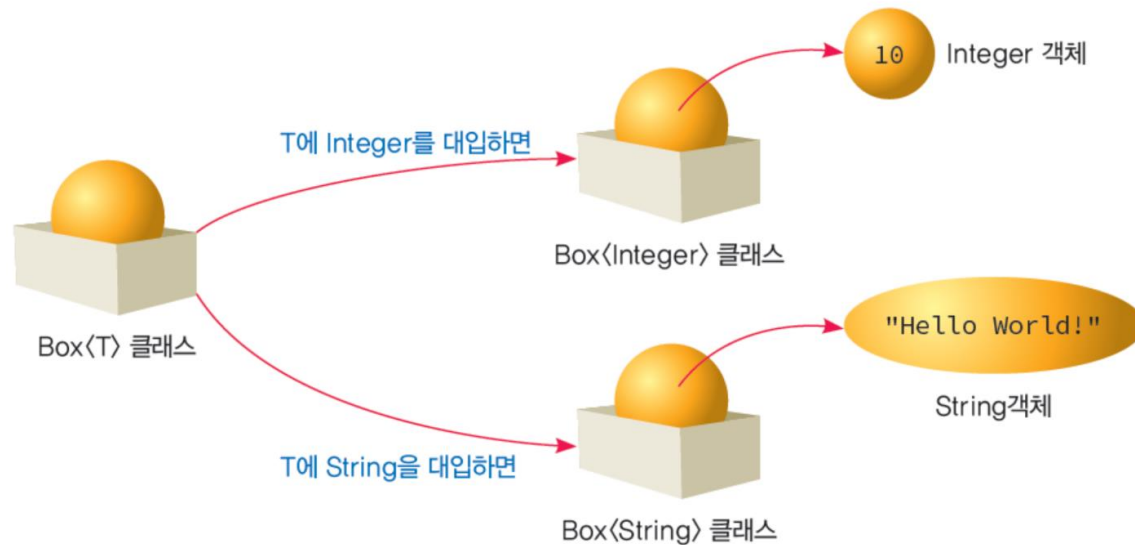
메서드나 필드에 필요한 타입을 타입 매개변수로 나타낸다.

```
}
```

타입 매개변수	설명
E	원소(Element)
K	키(Key)
N	숫자(Number)
T	타입(Type)
V	값(Value)

# Quiz 1 다른 정답

```
public class Box0<T> {  
    private T data;  
    public void set(T data) { this.data = data; }  
    public T get() { return data; }  
}
```



# ArrayList Review

- 가변 크기의 배열
- 객체를 저장할 수 있는 배열
- Java 내 collection 중 하나 (collection : 자바에서 자료구조를 구현한 클래스)
- 아래 표는 주요 메서드 설명

메서드	설명
<code>boolean add(E e)</code>	객체를 맨 끝에 추가한다.
<code>void clear( )</code>	저장된 모든 객체를 제거한다.
<code>boolean contains(Object o)</code>	명시한 객체의 저장 여부를 조사한다.
<code>boolean isEmpty( )</code>	리스트가 비어 있는지 조사한다.
<code>Iterator&lt;E&gt; iterator( )</code>	Iterator를 반환한다. • Iterator (반복자) : 컬렉션의 원소에 순차적으로 접근할 때 이용
<code>boolean remove(Object o)</code>	명시한 첫 번째 객체를 제거하고, 제거 여부를 반환한다.
<code>int size( )</code>	저장된 전체 객체의 개수를 반환한다.
<code>T[] toArray(T[] a)</code>	리스트를 배열로 반환한다.

# ArrayList 사용 예

(Exercise 4 - LoginProgram)

```
private ArrayList<User> users;  
private User user;  
public LoginProgram() {  
    users = new ArrayList<>();  
    users.add(new Admin("admin", "admin", users));  
    user = null;  
}
```

```
public void join(String id, String pw) {  
    User user = new User(id, pw);  
    users.add(user);  
    System.out.println(id + "님 회원 가입을 축하합니다.");  
}
```

```
public void withdraw() {  
    users.remove(user);  
    user = null;  
    System.out.println("탈퇴가 완료되었습니다.");  
    System.out.println("이용해 주셔서 감사합니다.");  
}
```

# Generic method

- 타입 매개변수를 사용하는 메서드
- 제네릭 클래스뿐만 아니라 일반 클래스의 메서드에서도 타입 매개변수를 사용해서 제네릭 메서드를 정의할 수 있음

```
<타입매개변수> 반환타입 메서드이름(...) {  
    ...  
}
```

2개 이상의 타입 매개변수도 가능하다.

# 연습문제 1

- practice package의 Practice1.java 를 에러가 나지 않도록 완성해 보자.

```
public static void main(String[] args) {  
    Integer[] iArray = { 10, 20, 30, 40, 50 };  
    Double[] dArray = { 1.1, 1.2, 1.3, 1.4, 1.5 };  
    Character[] cArray = { 'K', 'O', 'R', 'E', 'A' };  
    String[] sArray = { "C++", "C#", "JAVA" };  
  
    System.out.println(getLast(cArray));  
    System.out.println(getLast(sArray));  
  
    swap(iArray, 0, 3);  
    System.out.println(Arrays.toString(iArray));  
  
    swap(dArray, 2, 4);  
    System.out.println(Arrays.toString(dArray));  
}
```

```
A  
JAVA  
[40, 20, 30, 10, 50]  
[1.1, 1.2, 1.5, 1.4, 1.3]
```



# 연습문제 1 & Quiz 3

```
public static <T> T getLast(T[] array) {  
    return array[array.length - 1];  
}
```

2 usages

```
public static <T> void swap(T[] array, int i, int j) {  
    T tmp = array[i];  
    array[i] = array[j];  
    array[j] = tmp;  
}
```

swap에서 숫자로만 이루어진 배열만을 전달받게 하는 방법을 padlet에 답을 남겨보자

- swap을 아래처럼 바꾸고

```
public static <T extends Number> void swap(T[] array, int i, int j) {  
    T tmp = array[i];  
    array[i] = array[j];  
    array[j] = tmp;  
}
```

- main 함수에 아래 줄을 추가하면 compile error 발생

```
swap(cArray, 3, 2);
```

# 연습문제 2

- practice package의 Practice2.java 를 에러가 나지 않도록 완성해 보자.

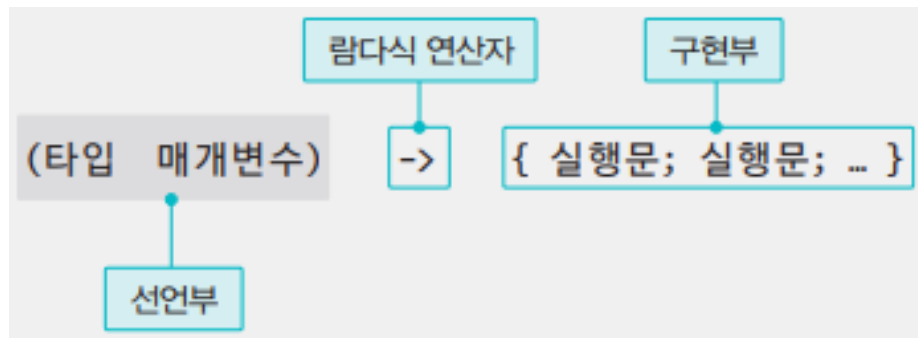
```
public class Practice2 {  
    public static void main(String[] args) {  
        Integer[] iArray = { 10, 20, 30, 40, 50 };  
        Double[] dArray = { 1.1, 1.2, 1.3, 1.4, 1.5 };  
  
        System.out.println(Arrays.toString(iArray) + "의 평균 : " + getAverage(iArray));  
        System.out.println(Arrays.toString(dArray) + "의 평균 : " + getAverage(dArray));  
    }  
}
```

```
[10, 20, 30, 40, 50]의 평균 : 30.0  
[1.1, 1.2, 1.3, 1.4, 1.5]의 평균 : 1.3
```

```
public static <T extends Number> double getAverage(T[] array) {  
    double result = 0;  
    for(T e : array)  
        result += e.doubleValue();  
    return result / array.length;  
}
```

# 람다식

- 메서드를 포함하는 익명 구현 객체를 전달할 수 있는 코드
- 나중에 실행될 목적으로 다른 곳에 전달될 수 있는 코드 블록
- 클래스를 생성하지 않고 함수를 객체로 간주
- 자바 8 부터 지원



```
int add(int x, int y) {  
    return x + y;  
}
```



```
(int x, int y) -> {  
    return x + y;  
};
```

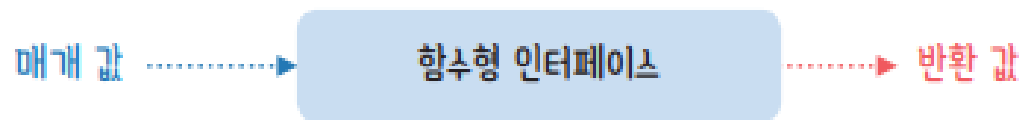
```
int max(int a, int b) {  
    return a > b ? a : b;  
}
```



```
(int a, int b) -> {  
    return a > b ? a : b;  
};
```

# 함수형 인터페이스

- 추상 메서드가 1개만 있는 인터페이스



- Comparator 인터페이스 (지난 주의 Comparable과는 다름)
  - 객체에 순서를 정하기 위하여 사용되는 함수형 인터페이스

```
Arrays.sort(strings, new Comparator<String>() {  
    public int compare(String first, String second) {  
        return first.length() - second.length();  
    }  
});
```

# 연습문제 3

- practice package의 Practice3.java 에서  
Arrays.sort의 두 번째 매개변수에 람다식을 이용하도록 바꿔보자

static <T> void

sort(T[] a, Comparator<? super T> c)

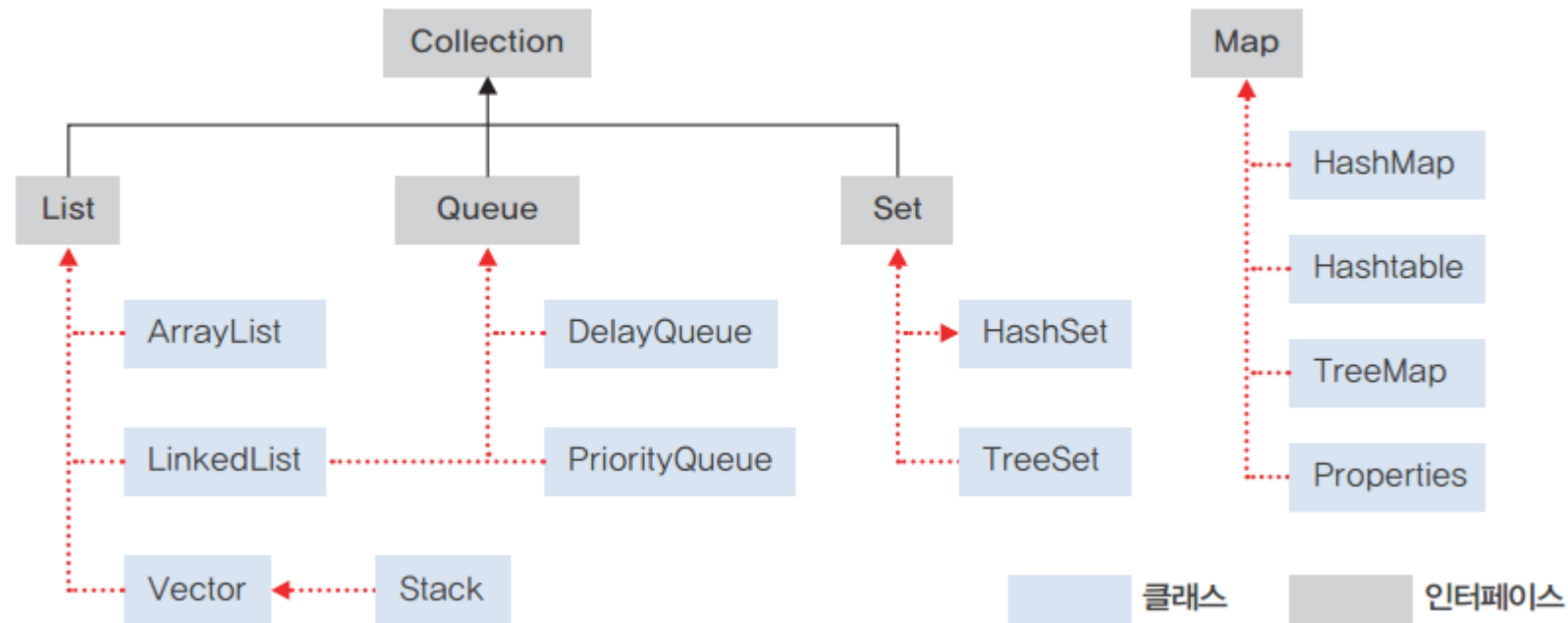
Sorts the specified array of objects according to the order induced by the specified comparator.

```
public static void main(String[] args) {  
    String[] strings = { "aaaaaaaaaaaaaaaaaaaa", "aaa", "aaaaaaaa" };  
  
    Arrays.sort(strings, new Comparator<String>() {  
        //  
        //     public int compare(String first, String second) {  
        //         return first.length() - second.length();  
        //     }  
        //  
    });  
  
    Arrays.sort(strings, (first, second) -> first.length() - second.length());  
  
    System.out.println(Arrays.toString(strings));  
}
```

[aaa, aaaaaaaa, aaaaaaaaaaaaaaaaaaaa]

# Collection


- 자바에서 자료구조를 구현한 인터페이스와 클래스
- 리스트(list), 스택(stack), 큐(queue), 집합(set), 해시 테이블(hash table)





# Collection

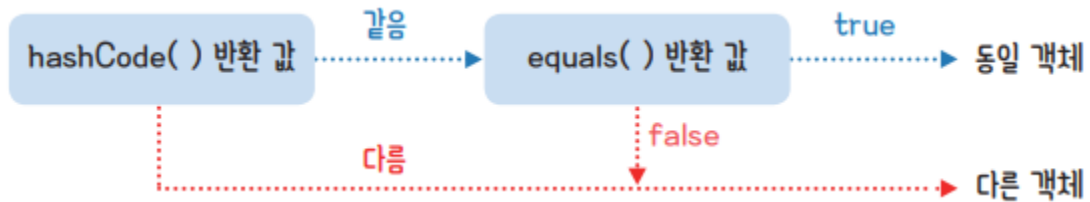
- Collection이 갖고 있는 주요 메서드

메서드	설명
<code>boolean add(E e)</code>	객체를 맨 끝에 추가한다.
<code>void clear()</code>	저장된 모든 객체를 제거한다.
<code>boolean contains(Object o)</code>	명시한 객체의 저장 여부를 조사한다.
<code>boolean isEmpty()</code>	리스트가 비어 있는지 조사한다.
<code>Iterator&lt;E&gt; iterator()</code>	Iterator를 반환한다. 
<code>boolean remove(Object o)</code>	명시한 첫 번째 객체를 제거하고, 제거 여부를 반환한다.
<code>int size()</code>	저장된 전체 객체의 개수를 반환한다.
<code>T[] toArray(T[] a)</code>	리스트를 배열로 반환한다.

메소드	설명
<code>hasNext()</code>	아직 방문하지 않은 원소가 있으면 true를 반환
<code>next()</code>	다음 원소를 반환
<code>remove()</code>	최근에 반환된 원소를 삭제한다.

# Set

- 원소의 중복을 허용하지 않음
- 포함된 원소에 순서가 존재하지 않음
- 객체의 중복을 확인하는 원리 □ hashCode()를 확인



# Set 인터페이스 구현

- HashSet
  - 해시 테이블에 원소를 저장하기 때문에 성능면에서 가장 우수
- TreeSet
  - 레드-블랙 트리(red-black tree)에 원소를 저장함. 따라서 값에 따라 순서가 결정되지만 HashSet보다는 느림
- LinkedHashSet
  - 해시 테이블과 연결 리스트를 결합한 것으로 원소들의 순서는 삽입했던 순서와 같음

# 연습문제 4

- practice package의 Practice4.java 에서 아래와 같은 출력이 되도록 완성해보자.

```
public static void main(String args[]) {  
    String[] fruits = {"사과", "바나나", "포도", "수박", "참외", "사과", "포도"};  
    Set<String> h = new HashSet<>();  
  
    // fruits 배열을 List인 list에 넣기  
    List<String> list = Arrays.asList(fruits);  
  
    //h에 list의 모든 요소를 넣기  
    h.addAll(list);  
  
    System.out.println("h 출력하기 : " + h);  
}
```

h 출력하기 : [포도, 수박, 사과, 참외, 바나나]

# 연습문제 5

- practice package의 Practice5.java 에서 아래와 같은 출력이 되도록 완성해보자.

```
public static void main(String args[]) {
    String[] fruits = {"사과", "바나나", "포도", "수박", "참외", "사과", "포도"};
    Set<Fruit> h = new HashSet<>();

    // 반복문과 fruits 배열을 이용해 h2에 Fruit 넣기
    for (String s : fruits)
        h.add(new Fruit(s));

    System.out.println("h의 크기는 : " + h.size());
    System.out.println("h 출력하기 : " + h);

    //h2에 있는 모든 요소 제거
    h.clear();

    //그냥 출력이 아니라, Hashmap이 갖고있는 메소드를 이용한 결과값 호출
    System.out.println("h가 비어 있나요? : " + h.isEmpty());
}
```

```
h의 크기는 : 5
h 출력하기 : [수박, 참외, 사과, 바나나, 포도]
h가 비어 있나요? : true
```

```

class Fruit {
    5 usages
    String name;
    1 usage
    public Fruit(String name) { this.name = name; }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Fruit fruit = (Fruit) o;
        return Objects.equals(name, fruit.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name);
    }

    @Override
    public String toString() { return name; }
}

```

- 오버라이딩 해야 의도대로 출력됨
- IntelliJ에서 단축키 alt + insert를 눌러 자동완성 가능

# Map

- 키와 값, 쌍으로 구성된 Entry 객체를 저장하는 자료구조
- 맵이 사용하는 키와 값도 모두 객체
- 키는 중복되지 않고 하나의 값에만 매핑되어 있으므로 키가 있다면 대응하는 값을 얻을 수 있음

메서드	설명
<code>void clear()</code>	모든 매핑을 삭제한다.
<code>boolean containsKey(Object key)</code>	주어진 키의 존재 여부를 반환한다.
<code>boolean containsValue(Object value)</code>	주어진 값의 존재 여부를 반환한다.
<code>Set&lt;Map.Entry&lt;K, V&gt;&gt; entrySet()</code>	모든 매핑을 Set 타입으로 반환한다. →
<code>V get(Object key)</code>	주어진 키에 해당하는 값을 반환한다.
<code>boolean isEmpty()</code>	컬렉션이 비어 있는지 여부를 반환한다.
<code>Set&lt;K&gt; keySet()</code>	모든 키를 Set 타입으로 반환한다.
<code>V put(K key, V value)</code>	주어진 키-값을 저장하고 값을 반환한다.
<code>V remove(Object key)</code>	키와 일치하는 원소를 삭제하고 값을 반환한다.
<code>int size()</code>	컬렉션의 크기를 반환한다.
<code>Collection&lt;V&gt; values()</code>	모든 값을 Collection 타입으로 반환한다.

- `Map.Entry<K, V>`가 갖고 있는 메서드

메서드	설명
<code>K getKey()</code>	원소에 해당하는 키를 반환한다.
<code>V getValue()</code>	원소에 해당하는 값을 반환한다.
<code>V setValue()</code>	원소의 값을 교체한다.

# Map 인터페이스 구현

- HashMap
  - 키와 값으로 null 이용 가능, 삽입, 검색 시간복잡도  $O(1)$
- Hashtable
  - Hashmap과 동일한 내부 구조이나 동기화(synchronize) 메서드로 구성되어 스레드에 안전
  - 키와 값으로 null 이용 불가
- TreeMap
  - 이진 트리 기반 Map 컬렉션, 객체 저장 시 자동으로 정렬됨
- Properties
  - Hashtable의 하위 클래스, 키와 값을 String 타입으로 제한



# 연습문제 6

- practice package의 Practice5.java 에서 아래와 같은 출력이 되도록 완성해보자.
- 프로그램 실행 후 맨 마지막 결과를 padlet에 남겨보자.

```
public static void main(String args[]) {
    HashMap<Integer, String> languages = new HashMap<>();

    // 원소 넣기 (1, "Java"), (2, "Python") (3, "JavaScript")
    languages.put(1, "Java");
    languages.put(2, "Python");
    languages.put(3, "JavaScript");

    System.out.println("languages 출력 : " + languages);

    // 원소 바꾸기 (1, "hello"), (2, "world")
    languages.put(1, "hello");
    languages.replace(2, "world");

    //entrySet()과 Entry<>를 이용한 key, value 출력
    for(Map.Entry<Integer, String> entry : languages.entrySet())
        System.out.println("키 : " + entry.getKey() + " 값 : " + entry.getValue());
}
```

```
// key 2에 해당하는 것 삭제
String removedValue = languages.remove(2);
System.out.println("삭제된 값은 " + removedValue + "이고, languages 출력 : " + languages);

// 방법 2
System.out.println("삭제된 값은 " + languages.remove(key: 2) + "이고, languages 출력 : " + languages);

// 아래 실행 결과를 예측하시오.
languages.remove(1, "Java");
System.out.println("languages 출력 : " + languages);
```

languages 출력 : {1=Java, 2=Python, 3=JavaScript}

키 : 1 값 : hello

키 : 2 값 : world

키 : 3 값 : JavaScript

삭제된 값은 world이고, languages 출력 : {1=hello, 3=JavaScript}

languages 출력 : {1=hello, 3=JavaScript}

# Collections Class

- 컬렉션을 다루는 다양한 메서드를 제공하는 java.util 패키지의 클래스

- 정렬(Sorting)

```
static void reverse(List list)
static void sort(List list)
static void sort(List list, Comparator c)

static Comparator reverseOrder()
static Comparator reverseOrder(Comparator c)
```

- 섞기(Shuffling)

```
static void rotate(List<?> list, int distance)

static void shuffle(List<?> list)
static void shuffle(List<?> list, Random r)
```

- 탐색(Searching)

```
static <T> int binarySearch(List<T> list, T key)
static <T> int binarySearch(List<T> list, T key, Comparator<T> c)
```

# 예시

- practice package의 Practice7.java 의 출력 결과를 padlet에 적어보자.

```
public static void main(String args[]) {  
    List<Integer> list = new ArrayList<>();  
    for(int i = 1; i <= 10; i++)  
        list.add(i);  
  
    System.out.println("4의 인덱스를 찾아보면 " + Collections.binarySearch(list, key: 4));  
    System.out.println("-1의 인덱스를 찾아보면 " + Collections.binarySearch(list, key: 0));  
    System.out.println("11의 인덱스를 찾아보면 " + Collections.binarySearch(list, key: 11));  
  
    System.out.println(list);  
    Collections.rotate(list, distance: 3);  
    System.out.println(list);  
    Collections.shuffle(list);  
    System.out.println(list);  
}
```

# Quiz

- 실행 예에서 search 음수 결과의 의미는 무엇일까? → -(추가된다면 들어갈 index) -1
- 아래 예에서 실행마다 출력값이 바뀌는 부분은 무엇일까? → shuffle은 매번 다르게 됨

4의 인덱스를 찾아보면 3

-1의 인덱스를 찾아보면 -1

11의 인덱스를 찾아보면 -11

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

[8, 9, 10, 1, 2, 3, 4, 5, 6, 7]

[4, 8, 2, 7, 9, 3, 5, 6, 10, 1]