# 2 학기
# JAVA Class

'이것이 자바다'

**2025.11**

**Cover Description**

made by Lewis

# Test.1

- 다음 조건의 클래스를 완성 하시오.

1. Animal 클래스
   멤버 변수 : name, age
   생성자 : name, age 를 받아서 저장
2. AnimalDoing 인터페이스
   먹는다, 짖는다, 걷는다, 앉는다, 뛴다, 문다.
3. Dog, Cat, Horse 클래스를 생성 하시오( Animal 상속, AnimalDoing 구현 해서 작성 하시오)
4. 실행클래스를 추가로 생성하고 다음과 같이 출력되도록 코딩 하시오.

```
바둑이 eat food.
바둑이 bark 멍멍.
바둑이 run.
나비 eat food.
나비 bark 야옹.
적토마 run.
적토마 bite.
```

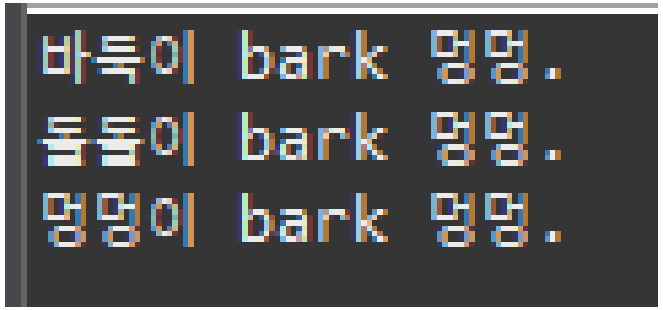
# Test.2

Test.1 에서 Dog 클래스를 배열로 선언하여 아래와 같이 출력되도록 코딩 하시오

```
멍멍이1 bark 멍멍.
멍멍이2 bark 멍멍.
멍멍이3 bark 멍멍.
멍멍이4 bark 멍멍.
멍멍이5 bark 멍멍.
```

# Test.3

Test.1 에서 Dog 클래스를 ArrayList로 선언하여 아래와 같이 출력되도록 코딩 하시오



01 자료구조

02 Array, ArrayLlist, Stack, Queue, Map

03 정렬 알고리즘

컴퓨터가 이해하는 코드는
누구라도 작성할 수 있습니다.
뛰어난 프로그래머는 사람이
이해하는 코드를 작성합니다.

# 자료 구조

- **자료구조란** : 자료를 효율적으로 저장하고 관리하는 방법
  - 전화번호부(이름으로 빠르게 찾기)
  - 책장(순서대로 정리)
  - 줄서기(먼저 온 사람이 먼저 나감)

➔ 빠르게 찾고
빠르게 추가하고
빠르게 삭제하기 위해

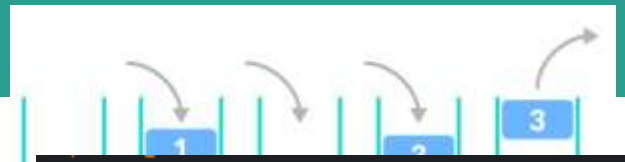
# 자료 구조

1. 배열 (Array)
   - 크기 고정, 인덱스로 접근
   - int[] number = new int[5];
     numbers[0] = 10;
     numbers[1] = 20;
     System.out.println(numbers[1]);

2. 리스트 (ArrayList)
   - 자동 크기 증가, 추가/삭제 쉬움, 인덱스로 접근
   - ArrayList<String> names = new ArrayList<>();
     names.add("Kim");
     names.add(“Ha");
     System.out.println(names.get(1));

# 자료 구조

3. 스택 (Stack)
   - 마지막 넣은
   - 접시 쌓기, 
   Stack<Intege
   stack.push(1
   stack.push(2
   System.out.p

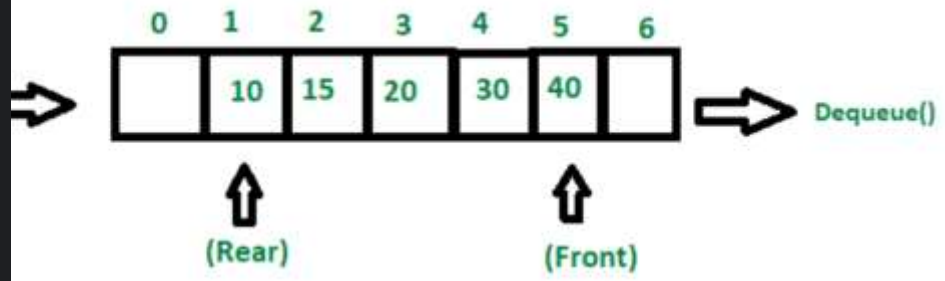



```
스택 내용: [10, 20, 30]
peek: 30
pop: 30
pop: 20
스택이 비었나요? false
남은 스택: [10]
```

```java
import java.util.Stack;
public class BaseStackTest {

    public static void main(String[] args) {
        // 1. 스택 생성
        Stack<Integer> stack = new Stack<>();

        // 2. push() – 데이터 넣기
        stack.push(10);
        stack.push(20);
        stack.push(30);

        System.out.println("스택 내용: " + stack);

        // 3. peek() – 맨 위 값 확인 (삭제 X)
        System.out.println("peek: " + stack.peek());   // 30

        // 4. pop() – 맨 위 값 꺼내기 (삭제됨)
        System.out.println("pop: " + stack.pop());     // 30

        // 5. pop() 한 번 더
        System.out.println("pop: " + stack.pop());     // 20

        // 6. 스택이 비었는지 확인
        System.out.println("스택이 비었나요? " + stack.isEmpty()); // false

        // 7. 남은 요소 출력
        System.out.println("남은 스택: " + stack);
    }
}
```

# 큐 구조

```java
import java.util.Queue;
import java.util.LinkedList;
public class BaseQueueTest {

    public static void main(String[] args) {
        // 1. Queue 생성 (LinkedList 사용)
        Queue<String> queue = new LinkedList<>();

        // 2. offer() - 데이터 추가
        queue.offer("홍길동");
        queue.offer("김철수");
        queue.offer("이영희");

        System.out.println("현재 큐: " + queue);

        // 3. peek() - 맨 앞 데이터 보기 (삭제 X)
        System.out.println("peek: " + queue.peek());  // 홍길동

        // 4. poll() - 맨 앞 데이터 꺼내기 (삭제됨)
        System.out.println("poll: " + queue.poll());  // 홍길동

        // 5. poll() 한 번 더
        System.out.println("poll: " + queue.poll());  // 김철수
```




; 라고 생성할 수 없음

```
현재 큐: [홍길동, 김철수, 이영희]
```

# 자료 구조

5. 맵 (Map)
   - **(Key → Value) 형태로 저장**
   - 주민번호 → 이름, 학생번호 → 성적 등

```
HashMap<String, Integer> map = new HashMap<>();
map.put("Kim", 90);
map.put("Lee", 85);
System.out.println(map.get("Kim"));
```

| key | value |
|---|---|
| firstName | Bugs |
| lastName | Bunny |
| location | Earth |

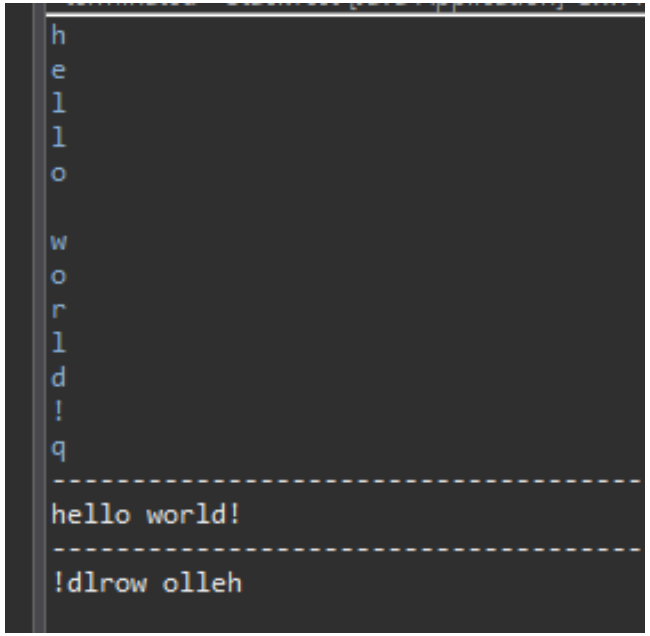
| | |
|---|---|
| put(key, value) | : key-value 데이터 저장 |
| get(key) | : key로 value 가져오기 |
| remove(key) | : key에 해당하는 데이터 삭제 |
| containsKey(key) | : key가 존재하는지 확인 |
| keySet() | : Map에 저장된 모든 key 목록 반환 |
| size() | : 데이터 개수 |

```
2 import java.util.HashMap;
3 import java.util.Map;
4 public class BaseMapTest {
```

```
홍길동 점수: 90
김철수 존재? true
모든 학생 목록:
홍길동 → 90
이영희 → 95
총 학생 수: 2
```

# Stack 사용 테스트 1

1. 문자열 입력을 하여 Hello World! 라는 입력 받아 화면에 표시한다.
2. 입력 받은 문자를 역순으로 출력하는 로직을 생성하고 뒤집힌 문자열을 출력한다.



# Stack 사용 테스트 2

1. Stack 역할을 하는 Stack2Cls class 를 개발 하시오
2. 길이는 10, isEmpty(), isFull(), push(int x), pop(), peek() 메소드를 구현 하시오
3. 실행 클래스를 생성하고 아래와 같이 출력되게 코딩 하시오

```
push: 10
push: 20
push: 30
peek: 30
pop: 30
pop: 20
Is it Empty?? false
```

# Queue 사용 테스트 1

1. Queue 를 활용하여 사람 3명을 순서대로 큐에 넣는다
   ( 철수, 경수, 정수)
2. poll() 을 사용하여 한 명씩 꺼내서 다음 형식으로 출력 하시오.
3. 처리 종료 후 “End!!” 메시지를 출력 하시오.

```
Next : 철수
Next : 경수
Next : 정수
End!!
```

# Map 사용 테스트 1

1. Map 을 활용하여 사람 3명과 총점을 순서대로 큐에 넣는다
   (홍길동:90 , 배철수 : 85, 강호동 : 95)
2. 배철수의 점수를 확인하고 이값이 있으면 데이터를 삭제한다.
3. 현재 등록된 사람과 총점을 확인하고 총 학생 목록과 총 학생 수를 출력한다.

# Queue 사용 테스트 2

1. 은행에 Customer 가 방문해서 방문 순서대로 업무를 등록한다.
2. Queue 를 활용하여 은행 업무 처리 프로그램을 만드시오.
3. Customer 클래스
   이름, 업무종류 를 생성자에서 받아서 저장
   ShowInfo메소드에서 업무 종류 출력
1. BankQueue 클래스 내부에서 Queue 를 사용
   addCustomer(Customer c) 메소드에서 Customer 추가
   nextCustomer() 메소드에서 Queue 값 가져와서 출력
1. BankMainEexc 클래스에서
   BankQueue 객체생성, Customer 5 명 생성 후 Queue 에 추가
   차례대로 업무 내용 출력

```
Register Customer: 홍길동
Register Customer: 김철수
Register Customer: 강감찬
Next Customer: 홍길동 (Job List: 예금)
Next Customer: 김철수 (Job List: 출금)
Next Customer: 강감찬 (Job List: 적금가입)
End of jobs
```

# 정렬 알고리즘

1. 버블 정렬
   - 가장 기본적인 정렬 알고리즘
   - 인접한 두 요소를 비교하여 위치를 교환(Swap)하는 작업을 반복하여 정렬하는 방식

   ex. [ 5, 1, 4, 2 ]
   1단계 : [ 1, 4, 2, 5 ]
   2단계 : [ 1, 2, 4, 5 ]

# 정렬 알고리즘

2. 선택 정렬
   - 가장 단순하고 직관적인 비교 기반 정렬 알고리즘
   - 가장 작은(혹은 큰) 데이터를 '선택'해서 정해진 위치로 보내는 방식
   - 매우 간단하다는 장점이 있지만, 데이터 양이 많아질수록 성능이 급격히 떨어짐

ex. [ 5, 3, 8, 1 ]

1단계 : 최소값 1을 찾아 맨 앞의 5와 교환 [ 1, 3, 8, 5 ]

2단계 : [3, 8, 5 ] 에서 최소 값 3 을 찾아 교환

3단계 : [ 8, 5 ] 에서 최소 값 5 를 찾아 교환 [ 5, 8 ]

➔ [ 1, 3, 5, 8 ]

# 정렬 알고리즘

3. 삽입 정렬
   - 이미 정렬된 부분 집합에 새로운 요소를 하나씩 가져와서 올바른 위치에 '끼워 넣는(Insert)' 방식으로 정렬
   - 삽입 정렬은 배열을 **정렬된 영역(왼쪽)**과 **정렬되지 않은 영역(오른쪽)**으로 나누어 진행
   - 삽입 정렬은 정렬된 배열에서 가장 빠른 성능을 보인다

# 정렬 알고리즘

4. 퀵 정렬
   - 가장 널리 사용되며 효율적인 정렬 알고리즘 중 하나로,
     분할 정복(Divide and Conquer) 방법을 사용
   - 배열 내에서 하나의 기준점(Pivot)을 정하고, 그 기준점을 중심으로 데이터를
     두 개의 작은 그룹으로 나눈 뒤(분할), 이 과정을 재귀적으로 반복하는 것

```java
import java.util.Queue;
import java.util.LinkedList;
public class BaseQueueTest {

    public static void main(String[] args) {
        // 1. Queue 생성 (LinkedList 사용)
        Queue<String> queue = new LinkedList<>();

        // 2. offer() – 데이터 추가
        queue.offer("홍길동");
        queue.offer("김철수");
        queue.offer("이영희");

        System.out.println("현재 큐: " + queue);

        // 3. peek() – 맨 앞 데이터 보기 (삭제 X)
        System.out.println("peek: " + queue.peek());  // 홍길동

        // 4. poll() – 맨 앞 데이터 꺼내기 (삭제됨)
        System.out.println("poll: " + queue.poll());  // 홍길동

        // 5. poll() 한 번 더
        System.out.println("poll: " + queue.poll());  // 김철수

        // 6. 큐가 비었는지 확인
        System.out.println("큐가 비었나요? " + queue.isEmpty());

        // 7. 남은 큐 출력
        System.out.println("남은 큐: " + queue);
    }
}
```

```
현재 큐: [홍길동, 김철수, 이영희]
peek: 홍길동
poll: 홍길동
poll: 김철수
큐가 비었나요? false
남은 큐: [이영희]
```

```java
import java.util.Stack;
public class BaseStackTest {

    public static void main(String[] args) {
        // 1. 스택 생성
        Stack<Integer> stack = new Stack<>();

        // 2. push() – 데이터 넣기
        stack.push(10);
        stack.push(20);
        stack.push(30);

        System.out.println("스택 내용: " + stack);

        // 3. peek() – 맨 위 값 확인 (삭제 X)
        System.out.println("peek: " + stack.peek());   // 30

        // 4. pop() – 맨 위 값 꺼내기 (삭제됨)
        System.out.println("pop: " + stack.pop());     // 30

        // 5. pop() 한 번 더
        System.out.println("pop: " + stack.pop());     // 20

        // 6. 스택이 비었는지 확인
        System.out.println("스택이 비었나요? " + stack.isEmpty()); // false

        // 7. 남은 요소 출력
        System.out.println("남은 스택: " + stack);
    }
}
```

```
스택 내용: [10, 20, 30]
peek: 30
pop: 30
pop: 20
스택이 비었나요? false
남은 스택: [10]
```