

JAVA 프로그램 실습

프로그램 따라하기

경북대학교
소프트웨어융합과
배희호 교수

객체 배열

- 객체 배열이란 객체들의 집합으로 구성된 배열을 말함
- 배열은 일련의 일정한 요소들을, 연속적이고, 순서적으로 저장하는 자료구조임
- 객체란 속성(변수)과 행동(함수)을 갖는 독립된 개체
- 객체 배열에서 각 요소는 객체를 참조하며, 이를 통해 복잡한 데이터 구조를 효율적으로 관리할 수 있음
- 예) 여러 사람의 정보를 저장하고 관리해야 하는 경우, 'Person'이라는 객체를 생성하고 이름, 나이, 주소와 같은 속성을 포함시킬 수 있음. 이렇게 정의된 'Person' 객체들을 배열에 저장하여, 필요에 따라 각 사람의 정보에 접근하고, 수정하고, 관리할 수 있음
- JAVA에서 동일 Type의 여러 객체를 생성하여 사용할 때는 객체 배열을 사용
 - 객체에 대한 Reference 배열임

객체 배열

- JAVA의 객체 배열 만들기 3단계
 - 배열 Reference 변수 선언
 - Reference 배열 생성
 - 배열의 각 원소 객체 생성

③ 객체 생성

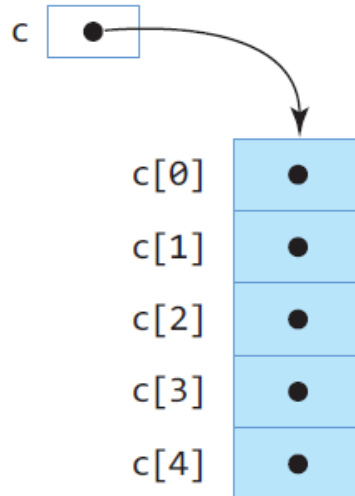
```
Circle[] c;
```

① 배열에 대한
레퍼런스 변수 선언

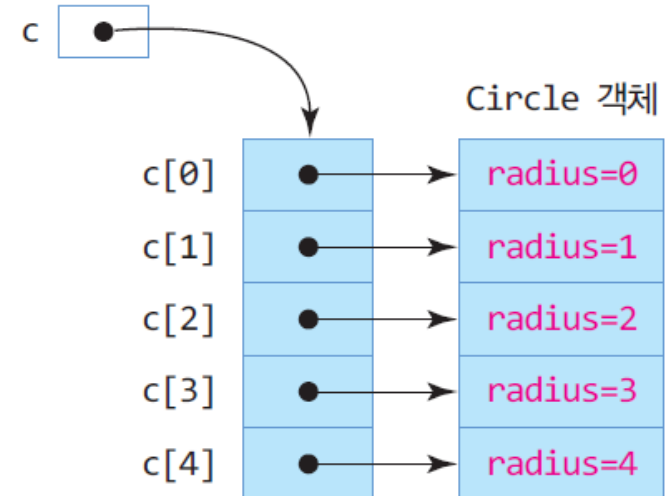


② 레퍼런스 배열 생성

```
c = new Circle[5];
```



```
for(int i=0; i<c.length; i++)  
    c[i] = new Circle(i);
```



객체 배열

- Class 또한 배열을 생성할 수 있음
- 객체 배열이 기본 Data Type 배열과 다른 점은 기본 Data Type으로 배열을 만들면 선언과 동시에 Memory가 생성되지만 객체 배열을 만들었을 때는 객체의 이름만 첨자 수만큼 생성
 - 실제 객체 내부의 Memory는 생성되지 않음

```
>>> Student 클래스 <<<  
class Student{  
    //...클래스의 내용  
}
```

```
>>> Student형의 객체 배열 생성<<<  
Student[] man = new Student[5];
```

객체 배열

- 각각의 참조 변수에 대한 Memory 생성 (생성자 호출)

```
man[0] = new Student();  
man[1] = new Student();  
man[2] = new Student();  
man[3] = new Student();  
man[4] = new Student();
```

객체 배열

```
Student[] student; // Student 배열에 대한 레퍼런스 변수 student 선언  
student = new Student[5]; // 레퍼런스 배열 생성
```

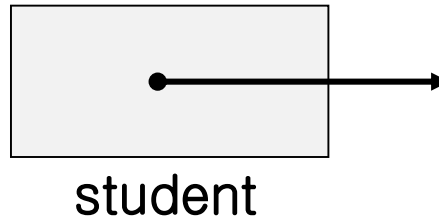
```
for (int i = 0; i < student.length; i++)  
    student[i] = new Student(); // 각 원소 객체 생성
```

```
Student[] student; // Student 배열에 대한 레퍼런스 변수 student 선언  
student = new Student[] { // 객체 배열 초기화  
    new Student(),  
    new Student(),  
    new Student(),  
    new Student(),  
    new Student()};
```

객체 배열

- 객체 배열 선언과 생성
 - 래퍼런스(참조) 변수 선언

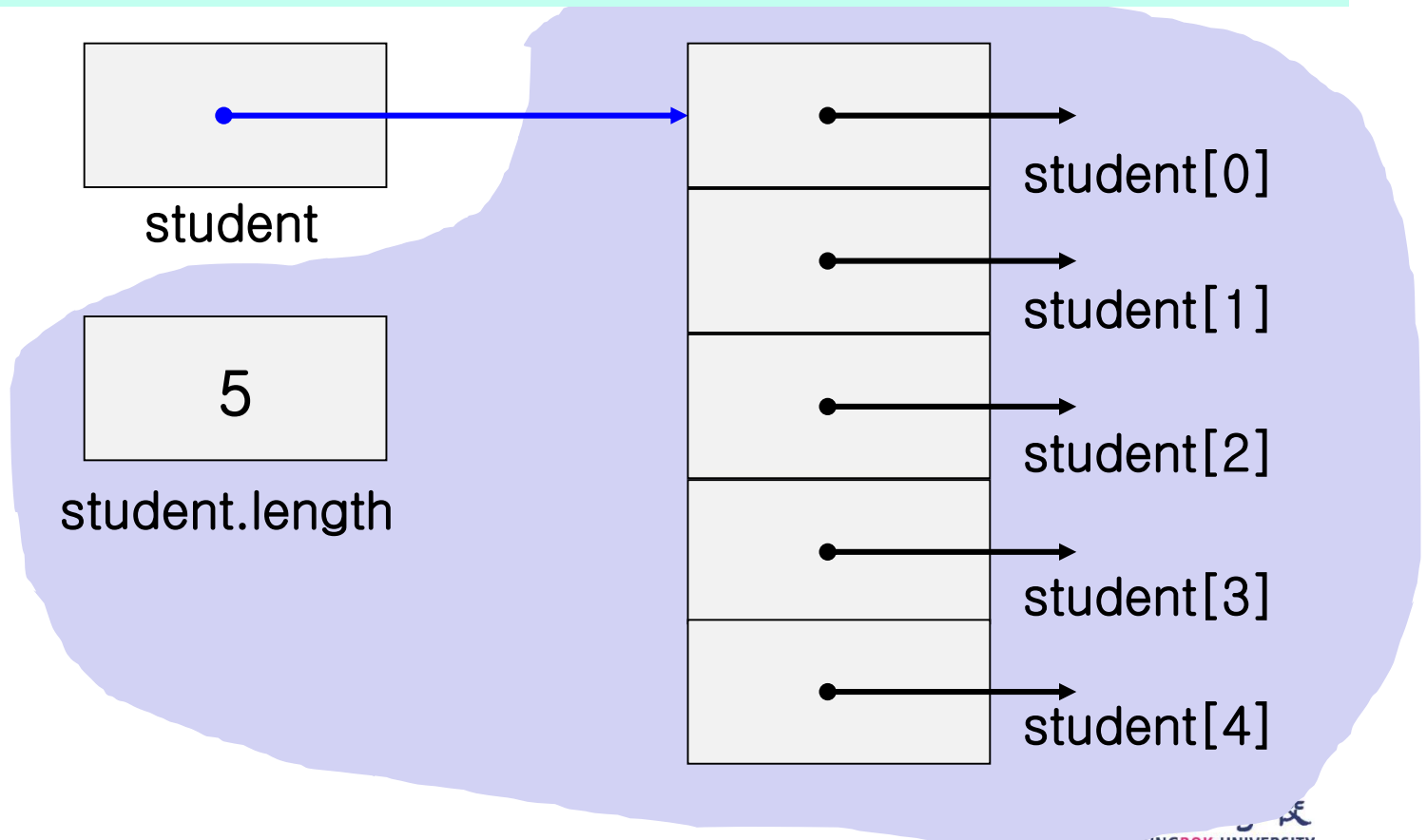
```
Student[] student;
```



객체 배열

- 객체 배열 선언과 생성
 - 레퍼런스 배열 생성

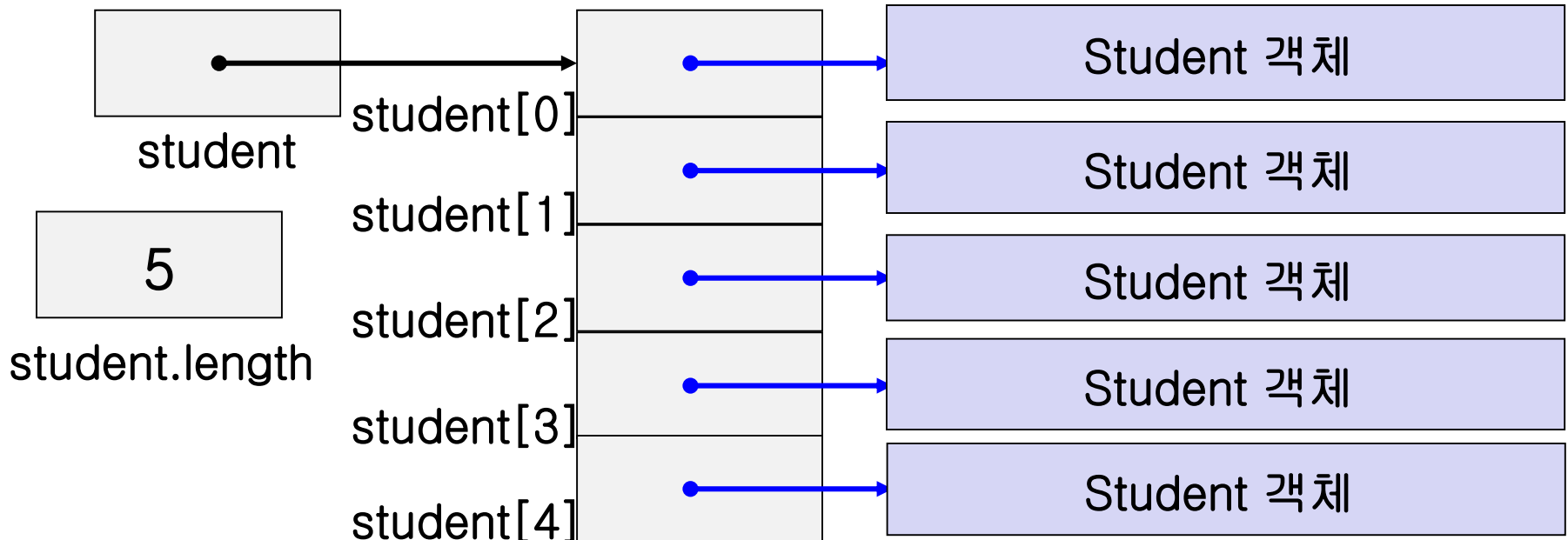
```
student = new Student[5];
```



객체 배열

- 객체 배열 선언과 생성
 - 객체 생성

```
for (int i = 0; i < student.length; i++)  
    student[i] = new Student();
```



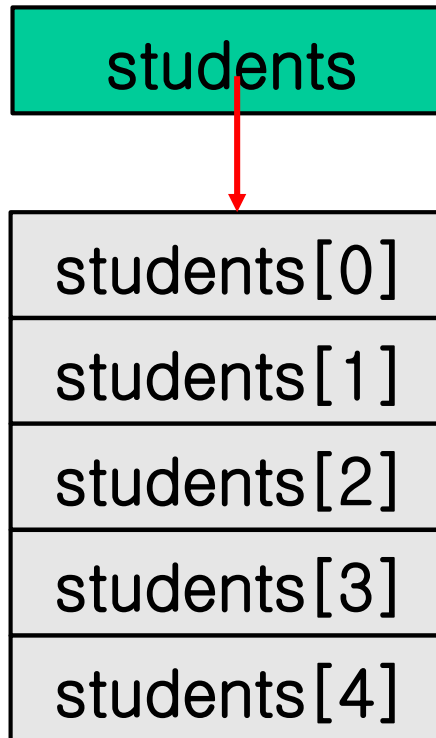
객체 배열 예제 1

- 5명의 학생의 번호(id), 이름(name), 성별(gender), 학점(grade)을 관리하는 Program을 만들어보자

번호	001	002	003	004	005
이름	홍길동	이대한	박찬호	한만국	홍미라
성별	남	남	남	남	여
학점	4.3	4.5	3.1	2.5	4.0

객체 배열 예제 1

■ 1차원 배열



객체 배열 예제 1

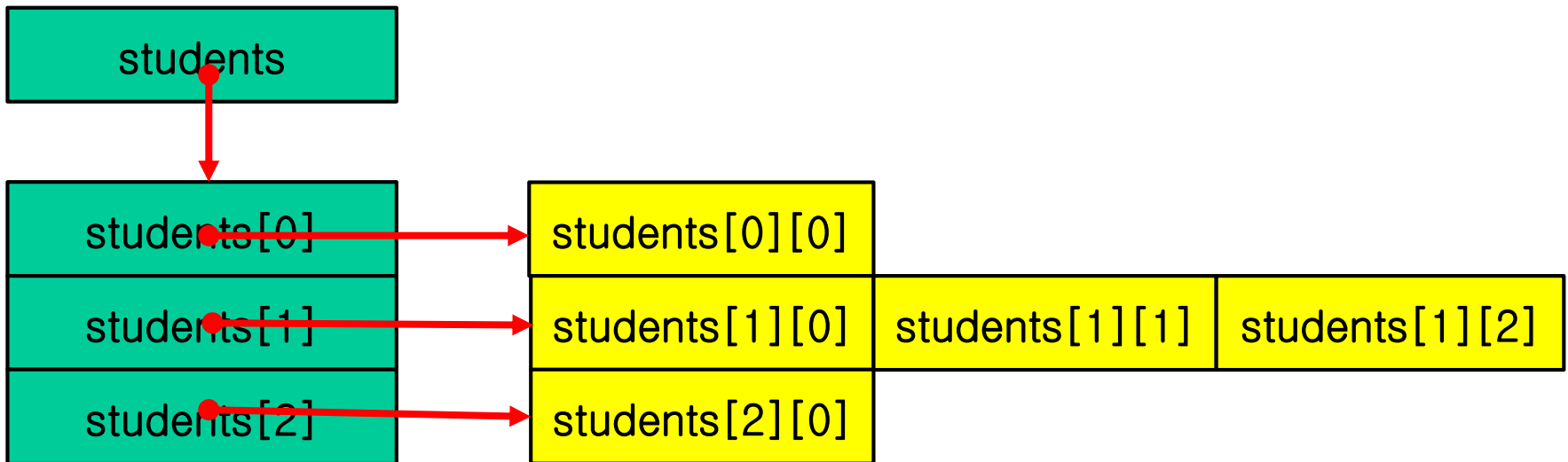
```
public class Student {  
    private String id;  
    private String name;  
    private String gender;  
    private double grade;  
  
    public Student(String id, String name, String gender, double grade) {  
        this.id = id;  
        this.name = name;  
        this.gender = gender;  
        this.grade = grade;  
    }  
    @Override  
    public String toString() {  
        return "id='" + id + 'W' + ", name='" + name + 'W' +  
            ", gender='" + gender + 'W' + ", grade=" + grade ;  
    }  
}
```

객체 배열 예제 1

```
public static void main(String[] args) {  
    Student[] students = new Student[5];  
    String[] name = {"홍길동", "이대한", "박찬호", "한민국", "홍미라"};  
    String[] gender = {"남", "남", "남", "남", "여"};  
  
    for (int i = 0; i < students.length; i++) {  
        students[i] = new Student("00" + (i + 1), name[i], gender[i] + "자",  
            (double) ((int) (new Random().nextDouble() * 4.5 * 100)) / 100);  
    }  
  
    for (int i = 0; i < students.length; i++) {  
        System.out.println("_____");  
        System.out.println(students[i]);  
    }  
}
```

객체 배열 예제 2

- 앞의 프로그램을 다음과 같은 구조의 다차원 객체 배열에 저장해보자



객체 배열 예제 2

```
public static void main(String[] args) {  
    Student[][] students = new Student[3][];  
    String[] name = {"홍길동", "이대한", "박찬호", "한민국", "홍미라"};  
    String[] gender = {"남", "남", "남", "남", "여"};  
  
    students[0] = new Student[1];  
    students[1] = new Student[3];  
    students[2] = new Student[1];  
  
    int count = 0;  
    for (int i = 0; i < students.length; i++) {  
        for (int j = 0; j < students[i].length; j++) {  
            students[i][j] = new Student("00" + (count+1), name[count],  
            gender[count] + "자",  
            (double) ((int) (new Random().nextDouble() * 4.5 * 100)) / 100);  
            count++;  
        }  
    }  
}
```

객체 배열 예제 2

```
for (int i = 0; i < students.length; i++) {  
    for (int j = 0; j < students[i].length; j++) {  
        System.out.println("_____");  
        System.out.println(students[i][j]);  
    }  
}  
}
```


Array

- Array 장점
 - 구현이 쉬움
 - 검색 성능이 좋음
 - Index를 이용한 무작위 접근(Random Access)이 가능하므로 검색에서 빠른 성능을 기대할 수 있음
 - 순차 접근(Sequential Access)의 경우에도 배열은 Data를 하나의 연속된 Memory 공간에 할당하므로 연결 리스트(Linked list)보다 빠른 성능을 보여줌
 - 참조를 위한 추가적인 Memory 할당이 필요 없음

Array

■ Array 단점

■ Data의 삽입과 삭제에 비효율적 임

- Data의 삽입(Insert)과 삭제>Delete)시 다음 항목의 모든 요소를 이동시켜야 함 (이를 연산)

- 작업이 수행되어 비효율적이며 Data의 수가 많아지면 비례하여 성능이 떨어지게 됨

■ 배열의 크기를 바꿀 수 없음

- 배열은 생성할 때 지정한 크기를 바꿀 수 없기 때문에 너무 크게 잡으면 Memory가 낭비되고 너무 작게 할당하면 그 이상의 Data를 저장할 수 없게 됨

Array

- Array 단점

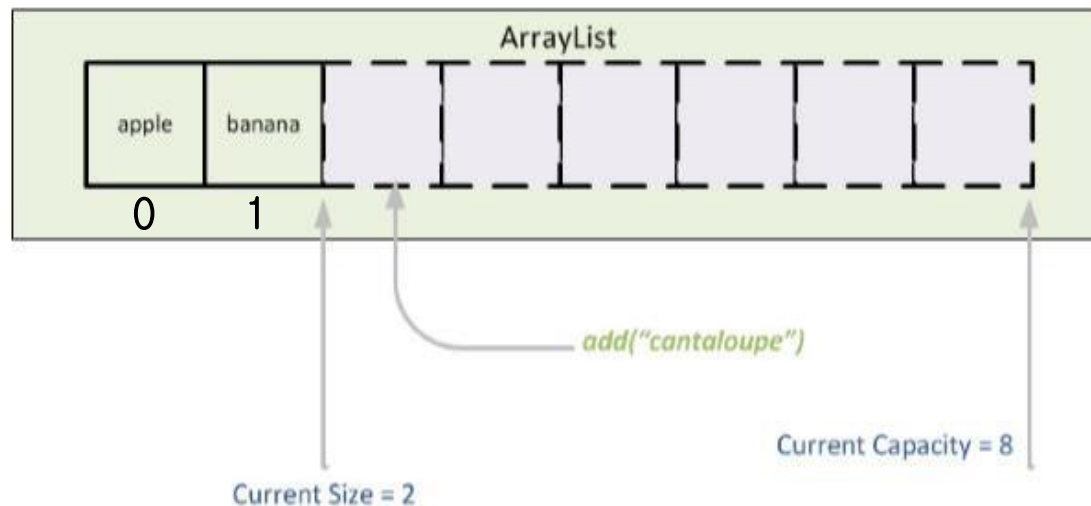
- Memory의 재사용이 불가능

- 배열은 초기 정의 시 크기만큼의 Memory를 할당 받아 사용하기 때문에 Data의 존재 유무와 상관없이 일정한 크기의 Memory 공간을 점유하고 있음

- 즉 이미 삭제한 Data라고 하더라도(배열 요소를 삭제) 배열 자체가 Memory에서 제거되지 않는 이상 삭제된 Data의 Memory 공간을 재사용 할 수 없음

ArrayList

- ArrayList는 JAVA에서 기본적으로 많이 사용되는 클래스
- ArrayList는 JAVA의 List 인터페이스를 상속받은 여러 클래스 중 하나임
- 일반 Array와 동일하게 동일한 크기의 연속된 Memory 공간을 사용하며 index는 0부터 시작



ArrayList

- Array와의 차이점은 Array의 크기가 고정인 반면 ArrayList는 크기가 가변적으로 변함
- 내부적으로 저장 가능한 Memory 용량(Capacity)이 있으며 현재 사용 중인 공간의 크기(Size)가 있음
- 만약 현재 가용량(Capacity) 이상을 저장하려고 할 때 더 큰 공간의 Memory를 새롭게 할당함

ArrayList

- ArrayList 값 추가하기
 - ArrayList의 값을 추가하기 위해서는 add() 메소드 사용
 - add(Object)
 - ArrayList의 마지막에 Data를 추가
 - add(int index, Object)
 - ArrayList의 index에 Data를 추가
 - Index 중간에 값을 추가하면 해당 Index부터 마지막 Index까지 모두 1씩 뒤로 밀려남
- ArrayList 값 변경하기
 - ArrayList 값 변경은 set() 메소드 사용
 - set(int index, Object)
 - set()을 사용하기 위해서는 바꾸려면 Data의 위치 Index를 알아야 변경이 가능

ArrayList

- ArrayList 값 삭제하기
 - ArrayList 값을 삭제하는 방법에는 remove()와 clear()가 있음
 - clear()
 - ArrayList의 모든 값을 삭제할 때 사용
 - remove()
 - 값을 하나씩 제거할 때 사용
 - remove(Object)
 - Object를 파라미터로 넘기는 경우 해당 ArrayList의 Object와 같은 값을 삭제
 - 같은 값이 여러 개인 경우 첫번째 같은 값을 제거
 - remove(int index)
 - ArrayList의 index에 해당하는 값을 삭제
 - 특정 Index의 객체를 제거하면 바로 뒤 Index부터 마지막 Index까지 모두 앞으로 1씩 당겨

ArrayList

- ArrayList 크기 구하기
 - ArrayList의 크기를 구하는 방법은 size() 메소드 사용
- ArrayList 값 출력하기
 - ArrayList의 결과를 출력하는 방법에는 get(int Index) 메소드 사용
 - Index를 입력하면 해당 Index의 Data가 출력
 - 전부 출력하고 싶다면 for문과 향상된 for문을 사용하여 출력을 할 수 있음
 - 추가로 Iterator를 사용하여 출력을 할 수도 있음

ArrayList

■ ArrayList 값 검색

- ArrayList에서 찾고자 하는 값을 검색하려면 ArrayList의 contains(Object) 메소드를 사용
- 만약 값이 있다면 true가 반환되고, 값이 없다면 false가 반환
- 값이 있는 Index를 찾으려면 indexOf(Object) 메소드를 사용
- 값이 있으면 Index를 반환하고, 만약 값이 없다면 -1을 반환

ArrayList

- java.util에서 제공하는 다수의 Data를 인덱스(index)를 통해 그룹 형태로 저장하는 객체
- **순서가 있으며 중복된 값을 저장**
- 제네릭 <>을 사용하여 리스트(list)의 Data Type을 지정할 수 있음
- Collection에서 제공하는 기능 중 하나로 다양한 형태, 수량의 Data를 저장할 수 있음

ArrayList

- ArrayList(List)를 초기화하는 방법
 - Arrays.asList()로 ArrayList 초기화
 - List.of()로 ArrayList 초기화
 - Double Brace Initialization을 이용하여 ArrayList 초기화
 - Stream으로 ArrayList 초기화

ArrayList

- Arrays.asList()로 ArrayList 초기화
 - Arrays.asList(array)는 인자로 전달된 배열을 List로 생성하여 반환
 - ArrayList 객체로 반환 받고 싶다면 new ArrayList<>(Arrays.asList(array))처럼 ArrayList로 변환하면 됨

```
public static void main(String[] args) {  
    List<String> list = Arrays.asList("apple", "grape", "banana", "kiwi");  
    System.out.println(list);  
    ArrayList<String> arrayList = new ArrayList<>(  
        Arrays.asList("apple", "grape", "banana", "kiwi"));  
    System.out.println(arrayList);  
}
```

ArrayList

- List.of()로 ArrayList 초기화
 - JAVA 9에서 List.of()가 추가되었으며, Arrays.asList()와 비슷한 방식으로 배열을 인자로 ArrayList를 초기화 가능
 - 기본적으로 List 객체로 반환되며, ArrayList로 반환 받려면 new ArrayList<String>()처럼 변환함

```
public static void main(String[] args) {  
    List<String> list = List.of("apple", "grape", "banana", "kiwi");  
    System.out.println(list);  
    ArrayList<String> arrayList = new ArrayList<>(  
        List.of("apple", "grape", "banana", "kiwi"));  
    System.out.println(arrayList);  
}
```

ArrayList

- Double Brace Initialization을 이용하여 ArrayList 초기화
 - Double Brace Initialization({{ ... }})을 이용하여 ArrayList를 초기화
 - 일반적으로 초기화 후 요소들을 추가하려면 아래처럼 add()를 각각 호출해야 함

```
public static void main(String[] args) {  
    ArrayList<String> list = new ArrayList<String>() {{  
        add("apple");  
        add("grape");  
        add("banana");  
        add("kiwi"); }};  
    System.out.println(list);  
}
```

ArrayList

- Stream으로 ArrayList 초기화
 - Stream을 사용하여 ArrayList를 초기화할 수 있음
 - 아래 코드는 String 배열의 값들을 ArrayList에 추가하는 예제

```
public static void main(String[] args) {  
    String[] arr = new String[] { "apple", "grape", "banana", "kiwi" };  
    ArrayList<String> list = new ArrayList<>(  
        Stream.of(arr) .collect(Collectors.toList()));  
    System.out.println(list);  
}
```

ArrayList

	Array	ArrayList
사이즈	초기화 시 고정 <code>int[] myArray = new int[6];</code>	초기화 시 사이즈를 표시하지 않음 (유동적) <code>ArrayList<Integer> myArrayList = new ArrayList<>();</code>
속도	초기화 시 메모리에 할당되어 속도가 빠르다	추가 시 메모리를 재할당하여 속도 가 느리다
변경	사이즈 변경 불가	추가 삭제 가능 <code>add()</code> , <code>remove()</code> 로 가능
다차원	가능 <code>int[][][] muttiArray = new int [3][3][3];</code>	불가능

객체 배열 예제 3

```
public static void main(String[] args) {  
    ArrayList<Student> students = new ArrayList<>();  
    String[] name = {"홍길동", "이대한", "박찬호", "한민국", "홍미라"};  
    String[] gender = {"남", "남", "남", "남", "여"};  
  
    for (int i = 0; i < name.length; i++) {  
        students.add(new Student("00" + (i + 1), name[i], gender[i] + "자",  
            (double) ((int) (new Random().nextDouble() * 4.5 * 100)) / 100));  
    }  
  
    for (int i = 0; i < students.size(); i++) {  
        System.out.println("_____");  
        System.out.println(students.get(i));  
    }  
}
```

객체 배열 예제 4

- 다음과 같이 회원(Member) 5명의 이름(name)과 나이(age) (20~25)를 초기화하는 프로그램을 작성해보자



객체 배열 예제 4

■ Member Class

```
public class Member {  
    private String name;  
    private int age;  
  
    public Member(){  
    }  
  
    public Member(String name){  
        this.name = name;  
    }  
    @Override  
    public String toString() {  
        return  
            "name = " + name +  
            ", age = " + age ;  
    }  
}
```

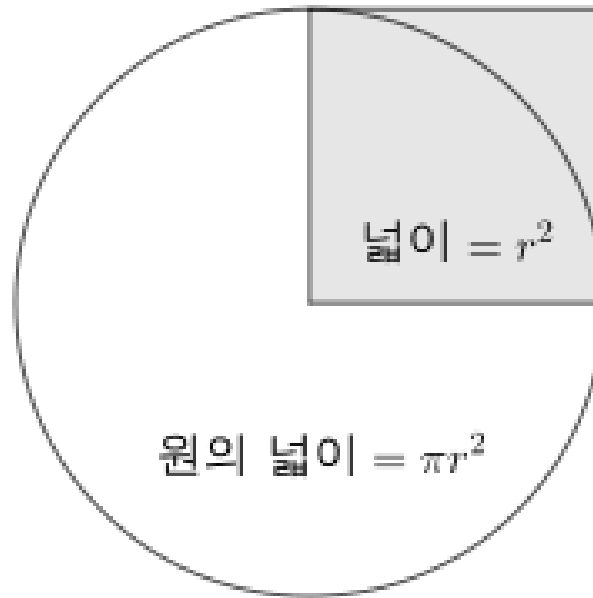
객체 배열 예제 4

■ Main Class

```
public static void main(String[] args) {  
    Member[] man = new Member[5];  
  
    man[0] = new Member("홍길동");  
    man[1] = new Member("이대한");  
    man[2] = new Member("한민국");  
    man[3] = new Member("나미라");  
    man[4] = new Member("고동욱");  
  
    for (int i = 0; i < man.length; i++) {  
        man[i].age = (int) ((Math.random() * (25 - 20 + 1)) + 20);  
    }  
    for (int i = 0; i < man.length; i++) {  
        System.out.println(man[i]);  
    }  
}
```

객체 배열 예제 5

- 원(Circle) 객체 5개를 가지는 배열을 생성하고, Circle 객체의 반지름(radius)을 0에서 4까지 각각 지정한 후, 면적을 출력하라.



객체 배열 예제 5

■ Circle Class

```
class Circle {  
    int radius;  
  
    public Circle(int radius) {  
        this.radius = radius;  
    }  
  
    public double getArea() {  
        return 3.141592 * radius * radius;  
    }  
}
```

객체 배열 예제 5

■ Main Class

```
public static void main(String[] args) {  
    Circle[] circles;  
    circles = new Circle[5];    // 기억 장소가 만들어지지 않음  
  
    for(int i = 0; i < circles.length; i++)  
        circles[i] = new Circle(i);  
  
    for(int i = 0; i < circles.length; i++)  
        System.out.println("반지름 : " + i + "면적 : " + circles[i].getArea());  
}
```

```
반지름 : 0, 면적 : 0.0  
반지름 : 1, 면적 : 3.141592  
반지름 : 2, 면적 : 12.566368  
반지름 : 3, 면적 : 28.2743280000000004  
반지름 : 4, 면적 : 50.265472
```

객체 배열 만들기 연습

- Book 클래스를 활용하여 2개짜리 Book 객체 배열을 만들고, 사용자로부터 책의 제목(title)과 저자(author)를 입력받아 배열을 완성하라.



- ✓ 속성 (필드, 데이터, 속성변수, 객체변수)
 - 제목, 저자
- ✓ 기능
 - 데이터 입력하기, 데이터 출력하기

객체 배열 만들기 연습

■ Book Class

```
public class Book {  
    private String title;  
    private String author;  
  
    public Book(String title, String author) {  
        this.title = title;  
        this.author = author;  
    }  
  
    public void toString() {  
        return String.format("%20s, %10s", title, author);  
    }  
}
```

객체 배열 만들기 연습

■ Main Class

```
public static void main(String[] args) {  
    Scanner keyboard = new Scanner(System.in);  
    Book[] book = new Book[2]; // Book 배열 선언  
  
    for (int i = 0; i < book.length; i++) {  
        System.out.print("제목>>");  
        String title = keyboard.nextLine();  
        System.out.print("저자>>");  
        String author = keyboard.nextLine();  
        book[i] = new Book(title, author); // 배열 원소 객체 생성  
    }  
    for (int i = 0; i < book.length; i++)  
        System.out.println(book[i]);  
}
```