Java프로그래밍

3강. Java 기본 문법(2)
(교재 2장-2)

컴퓨터과학과 김희천 교수

한국방송통신대학교

오늘의 학습목차

1. 배열
2. 문자열
3. Scanner 클래스와 입출력
4. 클래스 정의 (교재 3장)

1. 배열

1) 배열

- 같은 자료형의 원소를 정해진 개수만큼 가지고 있는 객체
- 배열의 크기는 배열이 초기화 또는 생성될 때 정해짐
- 숫자 인덱스(첨자)를 사용하여 특정 원소를 다룸

배열의 선언

- 선언할 때는 크기를 지정할 수 없음
- 형식은 자료형[ ] 변수이름; 또는 자료형 변수이름[ ];
- 예
  - ✓ int[ ] a; int b[ ];
  - ✓ int[ ][ ] c; int d[ ][ ]; int[ ] e[ ];
  - ✓ int f[10]; // 오류

2) 배열의 초기화

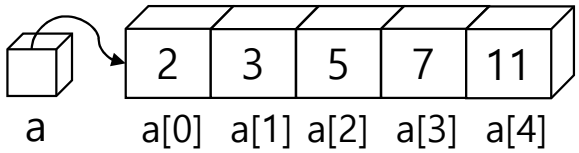
◆ 선언과 동시에 중괄호를 이용하여 초기값을 지정

✓ 자동으로 메모리 공간이 확보됨

✓ 초기화 또는 생성 과정을 거쳐야 배열의 원소를 사용할 수 있음

✓ 예

- int a[ ] = {2, 3, 5, 7, 11}; //선언과 동시에 초기화



- int anArray[ ][ ] = {{1, 2, 3}, {4, 5, 6}};
- int b[ ]; b = {4, 5, 6}; //오류

3) 배열의 생성

◆ 배열의 원소가 사용할 메모리 공간의 생성

✓ new 연산자를 이용

- 배열의 크기를 정하고 메모리 공간을 확보
- new 연산자는 메모리의 주소값을 리턴함
- 원소가 숫자인 경우 0, 참조형인 경우 null로 자동 초기화

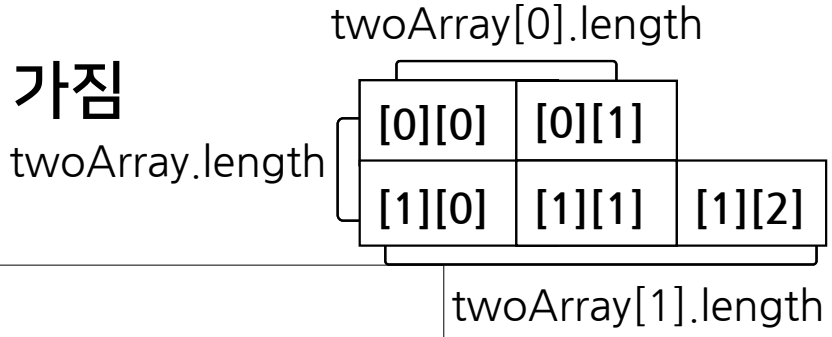
✓ 예

- int a[ ] = new int[3]; //선언과 생성
- int b[ ]; b = new int[10];
- int anArray4[ ][ ] = new int[3][2];

4) 배열의 크기

◆ 배열은 크기를 가지는 내장 속성 length를 가짐

✓ 사용법은 배열이름.length




```
public class ArrayTest {
   public static void main(String args[ ]) {
       int twoArray[ ][ ] = { {0, 1}, {10, 11, 12} };
        for(int i = 0; i<twoArray.length; i++)
              for(int j = 0; j<twoArray[i].length; j++)
        System.out.println( twoArray[i][j]);
   }
}
```

2. 문자열

1) String 클래스

- String 클래스는 문자열을 표현하고 처리하기 위한 참조형
- String 형의 변수는 참조형이나 기본형 변수처럼 사용할 수 있음

문자열 리터럴

- 이중 따옴표를 사용함
  - ✓ String s1 = “Java”;
  - ✓ String s2 = new String(“Java”); //생성자 사용
- 참조형 변수에는 null이라는 특별한 값을 지정할 수 있음
  - ✓ if (s1 != null) { … }

2) 문자열의 + 연산자

- 두 문자열을 연결하는 것
- (문자열 + 기본형) or (문자열 + 다른 참조형)도 가능
  - ✓ +연산자를 사용할 때, 기본형(또는 다른 참조형) 값은 문자열로 자동 형변환 가능
- print( )나 println( )에서 자주 사용됨
  - ✓ 1개 매개변수를 문자열로 바꾸어 출력함
  - ✓ System.out은 화면 출력을 위한 객체

```
System.out.println("result=" + " " + result);
System.out.println('A'+ 0); //65
System.out.println("A" + 0);
```

3. Scanner 클래스와 입출력

1) Scanner 클래스

- 키보드나 파일로부터 다양한 자료를 입력 받을 때 사용
  - ✓ 기본적으로 공백 문자로 구분되는 단어 단위로 입력됨
  - ✓ 문자열이나 기본형 값의 입력을 위해 nextXXX( ) 메소드를 제공함
- 클래스에 관한 설명을 보려면 아래 API 설명 페이지를 참조
  - ✓ http://docs.oracle.com/javase/8/docs/api/

키보드에서 입력을 받는 Scanner 객체

- System.in을 이용하여 Scanner 객체를 만들고 사용함

```
Scanner sc = new Scanner(System.in);
String name = sc.next( );
```

2) Scanner 클래스를 사용한 입력(1)

◆ Scanner 클래스의 입력용 메소드

- ✓ boolean hasNext( ) - 다음 단어가 있으면 true를 반환
  String next( ) – 단어를 읽어 String으로 반환
- ✓ boolean hasNextInt( ), int nextInt( )
- ✓ boolean hasNextDouble( ), double nextDouble( )
- ✓ boolean hasNextLine( ), String nextLine( );

```
Scanner s = new Scanner(System.in);
String name = s.next( );
```

2) Scanner 클래스를 사용한 입력(2)

```
import java.util.Scanner;

public class ScannerDemo2{
   public static void main(String args[]) {
      Scanner sc = new Scanner(System.in);
      while(sc.hasNextInt()) {
         System.out.println(sc.nextInt());
      }
  }
}
```

4. 클래스 정의(교재 3장)

1) 클래스 정의 문법

◆ 클래스 접근 제어자

✓생략, **public**, protected, private

✓ 또는 abstract, final

클래스 정의 문법

```
[접근 제어자] class 클래스이름
        [extends 부모클래스이름] [implements 인터페이스이름]
  {
    데이터필드 선언 …
    생성자 선언 …
    메소드 선언 …
  }
```

2) 클래스의 정의와 사용

◆ 클래스 정의

✓ 데이터 필드와 메소드를 정의
- 객체가 가지는 인스턴스 변수와 인스턴스 메소드
- 클래스가 가지는 클래스변수와 클래스 메소드

✓ 객체의 상태는 데이터 필드로, 행위는 메소드로 구현됨

✓ 메소드는 저장된 데이터를 이용해 기능을 수행

클래스의 사용

◆ 클래스형 변수를 선언할 때(클래스는 객체의 자료형)

◆ 객체를 생성할 때

◆ 상속받아 클래스를 정의할 때

3) 클래스의 접근 제어자(1)

- 클래스 접근 제어자의 의미
  - ✓ 클래스를 사용할 수 있는 범위를 제한하는 것
  - ✓ private과 protected는 특별한 경우에만 사용함

접근 제어자가 생략된 경우와 public class

- 클래스 선언에서 접근 제어자가 생략된 class
  - ✓ 같은 패키지에 있는 다른 클래스에서 사용 가능(패키지 접근 수준)
- public class로 선언된 경우
  - ✓ 모든 클래스에서 즉, 어디서나 사용 가능

3) 클래스의 접근 제어자(2)

```
class Circle {
    private double r;
    public Circle(double a) {
       r = a;
    }
    public double getArea( ) {
         return r * r * 3.14;
    }
     public double getRadius( ) {
         return r;
    }
}
```

```
public class CircleArea2 {
   public static void main(String args[ ]) {
     Circle c = new Circle(5);
     System.out.println(c.r); //오류
     System.out.println(c.getRadius());
      System.out.println(c.getArea( ));
  }
}
```

4) 데이터 필드의 접근 제어자(1)

- 클래스 정의에서 데이터 필드나 메소드를 정의할 때도 접근 제어자를 사용함
  - ✓ 데이터 필드를 사용할 수 있는 범위를 제한하는 것(정보 은닉)
  - ✓ 메소드의 접근 제어자도 의미가 같음

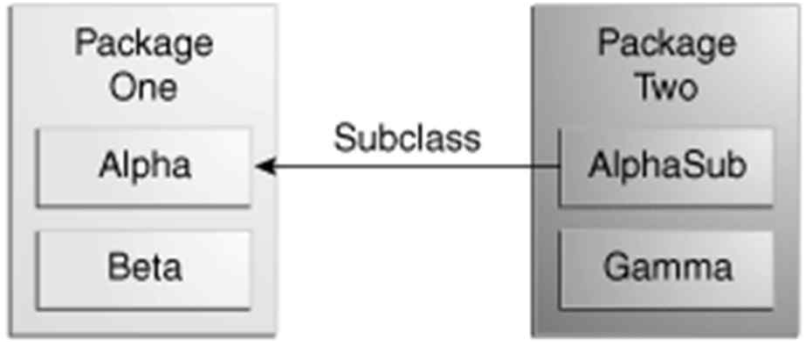
데이터 필드 접근제어자의 의미

- private 필드는 같은 클래스에서만 사용 가능
- 접근 제어자가 생략된 필드는 같은 패키지에 있는 다른 클래스에서 사용 가능
- protected 필드는 같은 패키지와 자식 클래스에서 사용 가능
- public 필드는 모든 클래스에서 사용 가능

4) 데이터 필드의 접근 제어자(2)

◆ 아래 4개의 클래스에서 Alpha 클래스의 어떤 멤버를 사용할 수 있는가?

Alpha 클래스에서 멤버의 선언	Alpha	Beta (같은 패키지)	AlphaSub (자식)	Gamma (다른 패키지)
public 멤버	사용가능	사용가능	사용가능	사용가능
protected 멤버	사용가능	사용가능	사용가능	
생략된 경우	사용가능	사용가능		
private 멤버	사용가능			



Java프로그래밍

다음시간안내

4강. 클래스와 상속