Java프로그래밍

2강. Java 기본 문법(1)
(교재2장-1)

컴퓨터과학과 김희천 교수

한국방송통신대학교

오늘의 학습목차

1. Java 프로그램과 기본 사항
2. 자료형
3. 연산자와 제어문

1. Java 프로그램과 기본 사항

1) Java 프로그램

◆ 성적 처리를 위한 Java 프로그램

```
class Grade {
  int e; //영어 성적을 위한 필드
  int m; //수학 성적을 위한 필드

  void input_grade(int a, int b)
  {
       e = a; //영어 성적 입력
       m = b; //수학 성적 입력
  }

  void output_grade( ) {
       //총점 출력
       System.out.println(e+m);
  }
}
```

```
public class GradeOutput {
  public static void main(String args[])  {
    Grade g1, g2; //성적을 표현하는 객체
    g1 = new Grade( );      // 객체 생성
    g2 = new Grade( );
    g1.input_grade(90, 85); // 성적 입력
    g2.input_grade(80, 80);

    g1.output_grade( );     // 총점 출력
    g2.output_grade( );
  }
}
```

2) 식별자(1)

- 클래스, 변수, 메소드, 레이블 등의 이름
- 프로그래머가 작명함

```
public class HelloApplication {
    static String szMsg = "Hello, Java!";
    public static void main(String args[]) {
        int nTest = 0;
        System.out.println(szMsg);
    }
}
```

작명 규칙

- 대소문자 구분
- 길이에 제한 없음
- 영 대소문자, 한글, 숫자, '_', '$'를 사용
- 숫자로 시작할 수 없음
- 키워드, true, false, null은 불가
- 잘못 작명된 예
  - ✓ 2002WorldCup, my#class
  - ✓ class, World Cup, lee@knou

2) 식별자(2)

◆ 식별자를 만들 때의 관례

✓ 클래스 : 첫 자는 대문자, 단어의 첫 글자는 대문자, 나머지는 소문자

- Car, HelloWorld, MyClass, String

✓ 메소드, 변수(필드) : 위와 같으나 첫 글자가 소문자

- speed, myCar, gearArea( )

✓ 상수 : 모든 문자를 대문자로 표기하고, 단어 사이에 '_'를 넣어 구분

- static final int NUM_GEARS = 6;

✓ 변수의 경우 자료형을 표시하기 위한 접두어를 붙이기도 함

- int nSpeed; String szStr1;

3) 키워드

◆ 키워드

✓ 의미가 미리 정해진 단어

✓ 프로그램에서 정해진 의미로만 사용해야 함

abstract	class	extends	import	private	switch	volitile
assert	const *	final	instanceof	protected	synchronized	while
boolean	continue	finally	int	public	this	
break	default	float	interface	return	throw	
byte	do	for	long	short	throws	
case	double	goto *	native	static	transient	
catch	else	if	new	strictfp	try	
char	enum	implements	package	super	void	

2. 자료형

1) 변수와 자료형

- 변수와 자료형
  - ✓ 변수를 선언할 때, 저장되는 값의 자료형을 선언
  - ✓ 메소드를 선언할 때, 반환 값의 자료형을 선언
  - ✓ 자료형에 따라 적용 가능한 연산이 다름
- 변수의 종류

인스턴스 변수	(클래스 정의에서 static이 아닌 필드) 객체가 소유하는 변수
클래스 변수	(클래스 정의에서 static 필드) 객체가 공유하는 변수
지역 변수	메소드 내부(또는 블록 내부)에서 선언된 변수
파라미터	메소드 호출 시 전달하는 값을 저장하기 위한 변수

2) 변수의 사용 범위

◆ **지역 변수와 파라미터**

✓ 선언된 곳부터 해당 블록이 종료될 때까지 유효함

- 메소드가 실행될 때 만들어지고 끝나면 없어짐

✓ 지역 변수는 초기값을 지정한 후 사용해야 함

✓ 지역 변수 선언에서 접근 제어자를 사용하지 않음

◆ 데이터 필드(인스턴스 변수 or 클래스 변수)

✓ 선언된 클래스 내부에서 사용 가능

✓ 클래스 외부에서의 사용 가능 여부는
접근 제어자(access modifier)에 따라 다름

✓ 예: class Circle { protected int radius; … }

3) Java의 기본 자료형

◆ Java의 기본 자료형

분류	키워드	길이 (byte)	값의 범위
문자	char	2	'\u0000'~'\uFFFF'
논리	boolean	1	true 또는 false
정수	byte	1	-128~127
	short	2	-32768~32767
	int	4	-2^{31}~2^{31}-1
	long	8	-2^{63}~2^{63}-1
실수	float	4	(+/-)약1.4E-45~3.4E38
	double	8	(+/-)약4.9E-324~1.8E308

4) 리터럴(1)

◆ 상수
- ✓ 리터럴(실제 데이터 값) 또는 값이 변하지 않는 변수
- ✓ final int nConst = 3; //선언 시 초기 값을 지정

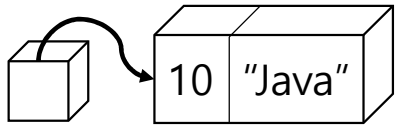
◆ 정수형 리터럴
- ✓ byte, short, int, long
- ✓ 소문자 l이나 대문자 L로 끝나면 long형, 나머지는 int
  - byte와 short는 허용 범위 안에서 int와 호환됨
- ✓ 26L, 26, 0b11010(2진수), 032(8진수), 0x1a(16진수)

4) 리터럴(2)

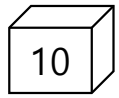
- 실수형 리터럴
  - ✓ 소수점이 있는 숫자
  - ✓ f나 F로 끝나면 float형, 나머지는 double형
  - ✓ 123.4f, 123.4, 1.234e2
- 문자형 리터럴
  - ✓ 1개의 문자를 표현하고 16비트 UNICODE로 인코딩됨
  - ✓ 단일 따옴표를 사용하고 Unicode 사용 가능
    - '\u0000'~ '\uFFFF'
    - (0~65536)의 수와 호환됨
  - ✓ (char)65, 'A', '\u0041', '가', '\uAC00'

5) 참조형

- 기본형을 제외한 모든 자료형
  - ✓ 참조 값(주소)을 가지는 자료형
- 배열, 클래스 형 등
  - ✓ int anArray[ ];
  - ✓ Circle myCircle;
  - ✓ String szStr;



- 참조형 변수는 저장 공간에 참조 값을 저장함
  - ✓ 실제 데이터는 별도의 공간에 저장됨
- 기본형 변수는 저장 공간에 값 자체를 저장함



6) 형변환

- 묵시적 자료형의 변환
  - ✓ 작은 타입에서 큰 타입으로는 자동 형변환
    - byte -> short -> int -> long -> float -> double
    - 개발자 -> 직원 -> 인간 -> 포유류
  - ✓ 예
    - double d = 5; //대입문
    - System.out.println("j=" + 10); //수식
    - double res = Math.sqrt(2); //메소드 호출시 인자의 유형
- 명시적 자료형의 변환
  - ✓ 큰 타입에서 작은 타입으로 변환할 때는 명시적 형변환이 필요
    - 문법은 **(자료형)피연산자**
  - ✓ 예
    - float f = (float)5.5;

3. 연산자와 제어문

1) 연산자의 종류

구분	연산자 예
산술연산자	+ - * / % 단항연산자 + - ++ --
비교연산자	> >= < <= == != instanceof
논리연산자	&& || 단항 ! 삼항 ?: & | ^
비트연산자	비트논리 & | ^ ~ 비트이동 << >> >>>
대입연산자	= += -= *= /= %= &= ^= |= >>= <<= >>>=
형변환연산자	(자료형)
기타	[ ] ( ) .

2) 명령 행 매개 변수

◆ 명령 행 매개 변수

- ✓ 프로그램을 실행할 때 전달하는 인자
- ✓ main( ) 함수에 전달되는 인자
- ✓ 문자열로 전달됨

> java CommandInputTest Kim 123

```
public class CommandInputTest {
     public static void main(String args[ ]) {
            System.out.println(args[0]);
            int n = Integer.parseInt(args[1]);
            System.out.println(n);
     }
}
```

3) 문장의 종류

- 수식문
- 변수 선언문
- 제어문
- 기타
  - ✓ 블록문
    - { … }
  - ✓ 레이블문
    - *레이블* : 문장
  - ✓ 예외처리문
    - try-catch 문
  - ✓ 동기화문
    - synchronized 문

4) 제어문

◆ 제어문

✓ 프로그램의 실행은 기본적으로 위에서 아래로 순차 실행됨

✓ 제어문은 실행 흐름을 바꿈

◆ 제어문의 종류

선택문	조건에 따른 문장의 선택 if문, switch문
반복문	조건에 따른 문장의 반복 for문, while문, do-while문
점프문	분기문 return문, break문, continue문

5) 선택문(1)

- if문
  - ✓ if (boolean-수식) 문장
- if-else문
  - ✓ if (boolean-수식) 문장 else 문장
- if와 else의 짝짓기
  - ✓ else는 자기 짝이 없는 가장 가까운 if와 짝을 이룸

```
int a = 2;
int b = 2;
if (a == 1)
      if (b == 2)
          System.out.println("a was 1 and b was 2.");
else
          System.out.println("a wasn't 1.");
```

5) 선택문(2)

- switch문
  - ✓ 다중 선택 구조
  - ✓ case 조건은 정수(long형 제외)와 호환되거나 String 값
  - ✓ default는 생략 가능하며, 어떤 case에도 해당되는 않는 경우 매칭됨
  - ✓ 만족되는 case를 실행한 후, break문을 만날 때까지 계속 실행

```
switch(n) {
   case 10:    System.out.println("10입니다.");
               break;
   case 20:    case 30: System.out.println("20이거나 30입니다.");
               break;
   default:    System.out.println("모르겠습니다.");
               break;
}
```

6) 반복문

- for-each문
  - ✓ 개선된 for문
  - ✓ 배열이나 컬렉션의 원소들을 차례로 다룰 때 편리
  - ✓ 형식은 **for (변수선언 : 배열) { 문장 … }**

```
int[ ] arrayOfInts = {32, 87, 3, 589, 12, 1076, 2000, 8};
for (int element : arrayOfInts) {
            System.out.print(element + " ");
}
System.out.println();
```

7) 점프문(1)

- break문
  - ✓ break문을 포함하는 가장 가까운 switch문, for문, while문, do-while문의 실행을 끝냄
    - 반복문이나 switch문을 빠져나갈 때 사용
  - ✓ 형식은 break;
  - ✓ 레이블을 사용하여 특정 블록 또는 특정 반복문을 빠져나갈 수 있음
    - 중첩 for문에서 바깥 for문을 종료하는 경우
  - ✓ 이 경우 형식은 break *레이블*;
    - 반복문에 레이블을 지정하려면 *레이블*: 반복문

7) 점프문(2)

- continue문
  - ✓ 반복문 안에서 사용함
  - ✓ 가장 가까이 있는 반복문의 다음 반복을 위한 조건식으로 즉시 제어를 이동하기 위한 것
  - ✓ 형식은 continue;
  - ✓ 레이블을 사용하여 특정 반복문의 다음 반복으로 이동할 수 있음
    - 중첩 for문에서 바깥 for문의 다음 반복으로 갈 때
    - 이 경우 형식은 continue *레이블*;

Java프로그래밍

다음시간안내

3강. Java 기본 문법(2)