

제2장 자바 기본 프로그래밍

2018년1학기

컴퓨터공학과 김 명 교수

목차

- ◆ 자바 기본 프로그램 구조
- ◆ 식별자, 자바 키워드
- ◆ 데이터 타입
- ◆ 변수, 리터럴, 상수
- ◆ 데이터 형 변환
- ◆ 표준 입출력
- ◆ 식과 연산자
 - ◆ 산술, 논리, 비트, 시프트 연산
 - ◆ 비교, 대입, 증감 연산
- ◆ if 문, switch 문

자바 프로그램 구조 - 맛보기 예제

3

```
public class Hello2 {
```

```
    public static int sum(int m, int n) {  
        return m + n;  
    }
```

메소드

```
    // main( ) 메소드에서 실행 시작
```

```
    public static void main(String[] args) {  
        int i = 20;  
        int s;  
        char a;
```

```
        s = sum(i, 10);    // sum( ) 메소드 호출
```

```
        a = '?';
```

```
        System.out.println(a); // 문자 '?' 화면 출력
```

```
        System.out.println("Hello2"); // "Hello2" 문자열 화면 출력
```

```
        System.out.println(s); // 정수 s 값 화면 출력
```

```
    }
```

```
}
```

클래스

```
?  
Hello2  
30
```

메소드


맛보기 예제 설명 (1/5)

4

- 클래스 만들기
 - ▣ Hello2 이름의 클래스 선언
 - ▣ public으로 선언 : 다른 클래스에서도 접근 가능

```
public class Hello2 {  
}
```

public,
abstract, final 가능



- main() 메소드
 - ▣ public static void으로 선언해야 함
 - ▣ 자바 프로그램은 main() 메소드부터 실행 시작
 - ▣ String[] args 로 실행인자 (command line arguments)를 전달 받음

```
public static void main(String[ ] args) {  
}
```

맛보기 예제 설명 (2/5)

5

- 멤버 메소드 (클래스에 속한 함수)
 - ▣ 메소드 sum() 정의
 - ▣ 메소드는 클래스 내에서만 선언 가능

```
public static int sum(int m, int n) {  
    ...  
}
```

- 변수 선언
 - ▣ 메소드 내에서 선언된 변수는 지역변수
 - ▣ 지역 변수는 메소드 실행이 끝나면 저장 공간 반환

```
int i=20;  
int s;  
char a;
```

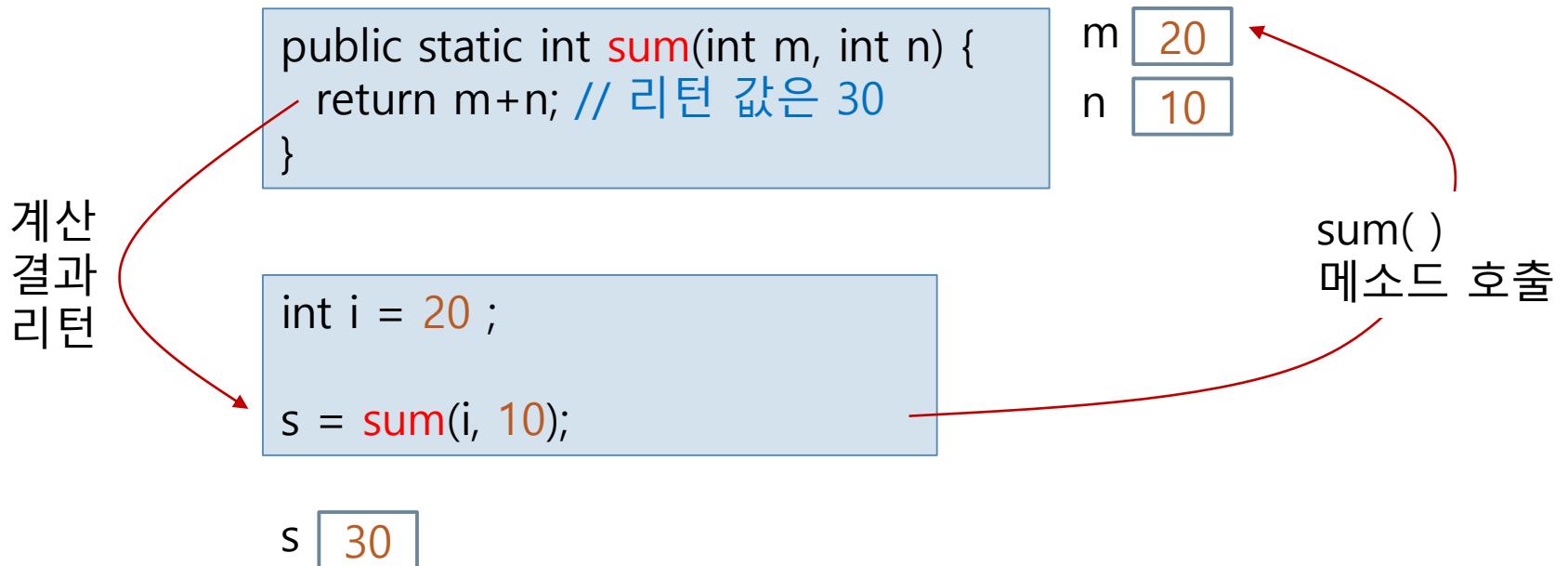
- 메소드 호출

```
s = sum(i, 10);    // 메소드 sum( ) 호출
```

맛보기 예제 설명 (3/5)

6

- sum() 메소드 호출과 리턴



맛보기 예제 설명 (4/5)

7

□ 화면 출력

- ▣ 표준 출력 스트림에 메시지 출력
- ▣ 표준 출력 스트림 System.out의 println 메소드 호출
- ▣ println : 여러 가지 데이터 타입 출력, 출력 후 다음 행으로 커서 이동

```
System.out.println(a);    // 문자 ? 를 화면에 출력
System.out.println("Hello2"); // "Hello2" 문자열을 화면에 출력
System.out.println(s);    // 정수 s의 값을 화면에 출력
```

▣ 참고

- System : java.lang 패키지에 속한 (final) 클래스
- out: 콘솔을 의미하는 PrintStream 객체
- println: 문자열을 출력하는 PrintStream 클래스의 메소드

맛보기 예제 설명 (5/5)

8

The image shows a screenshot of an IDE's project structure and class files. The project is named 'Hello2' and contains a 'src' directory with a 'default package' containing 'Hello1.java' and 'Hello2.java'. The 'JRE System Library [JavaSE-1.8]' is also visible, containing 'resources.jar' and 'rt.jar'. The 'rt.jar' is expanded to show the 'java.lang' package, which contains various classes including 'AbstractMethodError.class', 'AbstractStringBuilder.class', 'Appendable.class', 'ApplicationShutdownHooks.class', 'ArithmeticException.class', and 'ArrayIndexOutOfBoundsException.class'. The 'System.class' file is highlighted in the right-hand pane. Numbered annotations (1, 2, 3, 4) point to specific elements: 1 points to the 'Hello2.java' file, 2 points to the 'rt.jar' file, 3 points to the 'java.lang' package, and 4 points to the 'System.class' file.

- 1. Hello2
 - src
 - (default package)
 - Hello1.java
 - Hello2.java
 - JRE System Library [JavaSE-1.8]
 - resources.jar - C:\Program Files\Java\jre1.8\resources.jar
 - rt.jar - C:\Program Files\Java\jre1.8\rt.jar
 - com.oracle.net
 - com.oracle.nio
 -
 - java.beans.beancontext
 - java.io
 - java.lang
 - AbstractMethodError.class
 - AbstractStringBuilder.class
 - Appendable.class
 - ApplicationShutdownHooks.class
 - ArithmeticException.class
 - ArrayIndexOutOfBoundsException.class
- 2. String.class
- 3. StringBuffer.class
- 4. StringBuilder.class
- StringCoding.class
- StringIndexOutOfBoundsException.class
- SuppressWarnings.class
- System.class
- SystemClassLoaderAction.class

식별자 (이름, identifier)

9

- 식별자란?
 - ▣ 클래스, 변수, 상수, 메소드 등에 붙이는 이름

- 식별자의 원칙
 - ▣ 숫자, 영문자, '_', '\$' 로 구성됨.
 - ▣ 유니코드 문자 사용 가능, 한글 사용 가능
 - ▣ 첫 글자에 숫자를 사용할 수 없음.
 - ▣ 자바 언어의 키워드는 식별자로 사용할 수 없음.
 - ▣ 불린 리터럴 (**true**, **false**)과 널 리터럴(**null**)은 식별자로 사용할 수 없음
 - ▣ 길이 제한 없음

- 대소문자 구별
 - ▣ Test와 test는 별개의 식별자

식별자 이름 사례

10

□ 올바른 예제

```
int name;  
char student_ID;           // '_' 사용 가능  
void $func() { }           // '$' 사용 가능  
class Monster3 { }         // 숫자 사용 가능  
int whatisyourname;        // 길이 제한 없음  
int barChart; int barchart; // barChart와 barchart는 다름  
int 가격;                  // 한글 이름 사용 가능
```

□ 잘못된 예제

```
int 3Chapter;              // 첫 글자가 숫자  
class if { }               // if는 자바의 예약어  
char false;                // false 사용 불가  
void null() { }            // null 사용 불가  
class %calc { }            // '%' 는 특수 문자, 사용 불가
```

자바 키워드

11

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

식별자 이름 붙이는 관습

12

□ 클래스 이름

- 첫째 문자는 대문자
- 여러 단어로 구성된 경우는 각 단어의 첫째 문자만 대문자로 한다.

```
public class HelloWorld { }  
class Vehicle { }  
class AutoVendingMachine { }
```

□ 변수, 메소드 이름

- 첫 단어 이후 각 단어의 첫 번째 문자를 대문자로 시작한다.

```
int iAge;           // iAge의 i는 int의 i를 표시  
boolean blsSingle; // blsSingle의 처음 b는 boolean의 b를 표시  
String strName;     //strName의 str은 String의 str을 표시  
public int iGetAge() { } //iGetAge의 i는 int의 i를 표시
```

□ 상수 이름 : 모두 대문자로 표시

```
final static double PI = 3.141592;
```

자바의 데이터 타입

13

▣ 기본 타입 : 8 개

- boolean
- char
- byte
- short
- int
- long
- float
- double

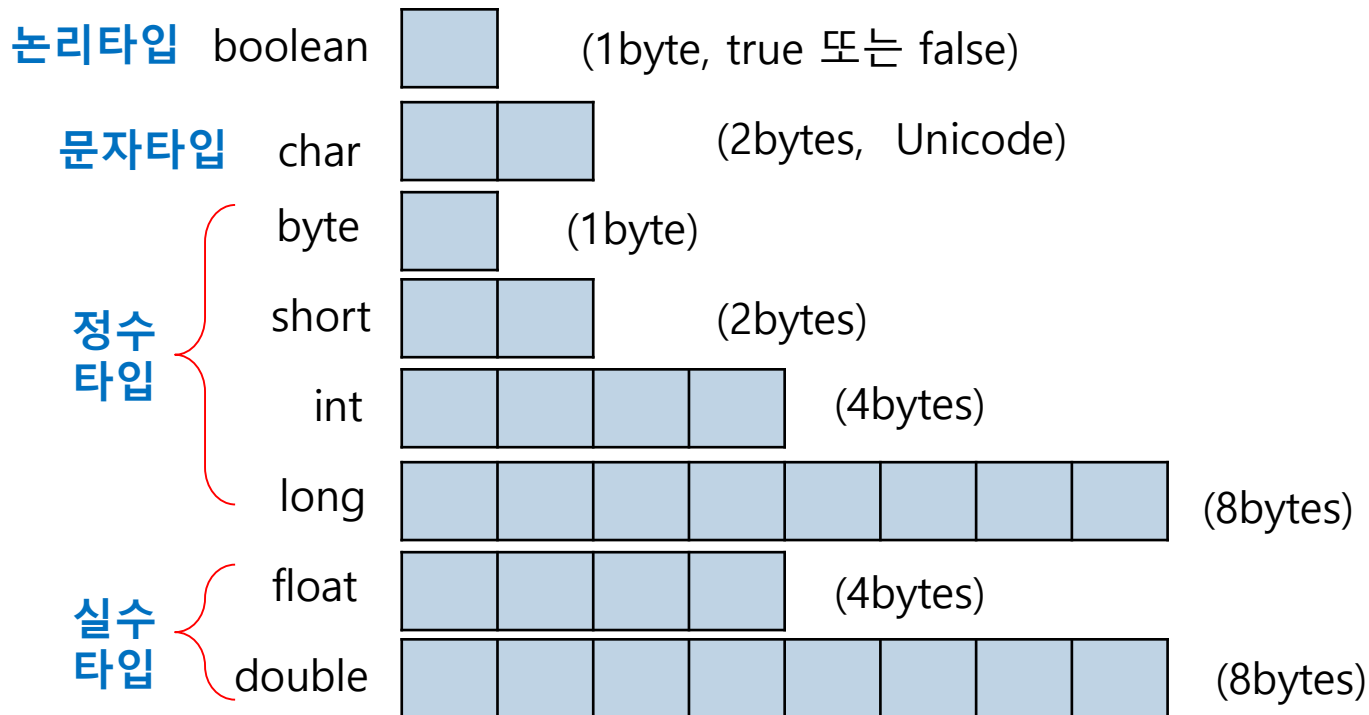
▣ 레퍼런스 타입 : 1개

- 클래스(class)에 대한 레퍼런스
- 인터페이스(interface)에 대한 레퍼런스
- 배열(array)에 대한 레퍼런스

자바의 기본 데이터 타입

14

- 기본 데이터 타입의 크기는 정해져 있으며, CPU나 운영체제에 따라 변하지 않음



변수와 선언

15

- 변수는 반드시 선언하고 초기화 한 후에 사용해야 한다.
- 선언의 위치는 변수를 사용하기 전에만 하면 된다.

```
int radius; // 선언  
Char c1 = 'a ', c2 = ' b ', c3 = ' c ' ; // 선언과 초기화  
Double weight = 75.56; // 선언과 초기화
```

```
sum = 0;  
for (int i = 0; i < 10; i++)  
    sum += i;
```

- 변수에 값 대입하는 명령문

```
radius = 10 * 5;  
c1 = 'r';  
weight = weight + 5.0;
```

리터럴과 정수 리터럴

16

- 리터럴(literal)
 - ▣ 프로그램에서 직접 표현한 값
 - ▣ 정수, 실수, 문자, 논리, 문자열 리터럴 있음
 - 사례) 34, 42.195, ' % ', true, " hello"
- 정수 리터럴 : 10진수, 8진수, 16진수, 2진수 리터럴

15 -> 10진수
015 -> 0으로 시작하면 8진수. 십진수로 13
0x15 -> 0x로 시작하면 16진수. 십진수로 21
0b0101 -> 0b로 시작하면 2진수. 십진수로 5



```
int n = 15;  
int m = 015;  
int k = 0x15;  
int b = 0b0101;
```

- ▣ 정수 리터럴은 int 형으로 컴파일
- ▣ long 타입 리터럴은 숫자 뒤에 L 또는 l을 붙여 표시
 - ex) long g = 24L;

실수 리터럴

17

- 소수점 형태나 지수 형태로 표현한 실수
 - 12. 12.0 .1234 0.1234 1234E-4

- 실수 타입 리터럴은 double 타입으로 컴파일

```
double d = 0.1234;
```

```
double e = 1234E-4; // 1234E-4 = 1234x10-4이므로 0.1234와 동일
```

- 숫자 뒤에 f(float)나 d(double)을 명시적으로 붙이기도 함

```
float f = 0.1234f;
```

```
double w = .1234D; // .1234D와 .1234는 동일
```

문자 리터럴

18

- 단일 인용부호(' ')로 문자 표현
 - ▣ 사례) 'w', 'A', '가', '*', '3', '글', \u0041
 - ▣ \u다음에 4자리 16진수 (2바이트 크기의 유니코드)
 - \u0041 -> 문자 'A'의 유니코드(0041)
 - \uae00 -> 한글문자 '글'의 유니코드(ae00)

```
char a = 'A';  
char b = '글';  
char c = \u0041; // 문자 'A'의 유니코드 값(0041) 사용  
char d = \uae00; // 문자 '글'의 유니코드 값(ae00) 사용
```

- 특수문자 리터럴은 백슬래시(\)로 시작
 - ▣ '\t' : tab '\n' : line feed
 - ▣ '\"' : single quote '\"' : double quote

논리타입 리터럴

19

- 논리값 표시
 - ▣ **true** 또는 **false** 뿐
 - ▣ Boolean 타입 변수에 대입 (저장) 하거나 조건문에 활용

boolean a = true;
boolean b = 10 > 0; // 10>0가 참이므로 b 값은 true
boolean **c = 1**; // 타입 불일치 오류. C/C++와 달리
// 자바에서는 1, 0을 참, 거짓으로 사용 불가

오류

while(**true**) { // 무한 루프. while(1)로 사용하면 안 됨
...
}

오류

기본 데이터 타입 이외의 리터럴

20

- **null** 리터럴
 - 레퍼런스에 대입
 - ~~int n = null;~~ // 기본 데이터 타입에는 사용 불가
 - String str = null;
- 문자열 리터럴
 - 이중 인용부호로 묶어서 표현
 - "Good", "Morning", "자바", "3.19", "26", "a"
 - 문자열 리터럴은 **String** 객체로 자동 처리

```
String str1 = "Welcome";  
String str2 = null;  
System.out.println(str1);
```

JDK7부터 숫자에 '_' 허용, 가독성 높임

21

- 숫자 리터럴의 아무 위치에나 ('_')를 허용함
 - ▣ 컴파일러는 리터럴에서 '_'를 빼고 처리함
- 사용 예

```
int price = 20_100;           // 20100과 동일
long cardNumber = 1234_5678_1357_9998L;
                        // 1234567813579998L와 같음
long controlBits = 0b10110100_01011011_10110011_111110000;
long maxLong = 0x7fff_ffff_ffff_ffffL;
int age = 2____5;           // 25와 동일
```

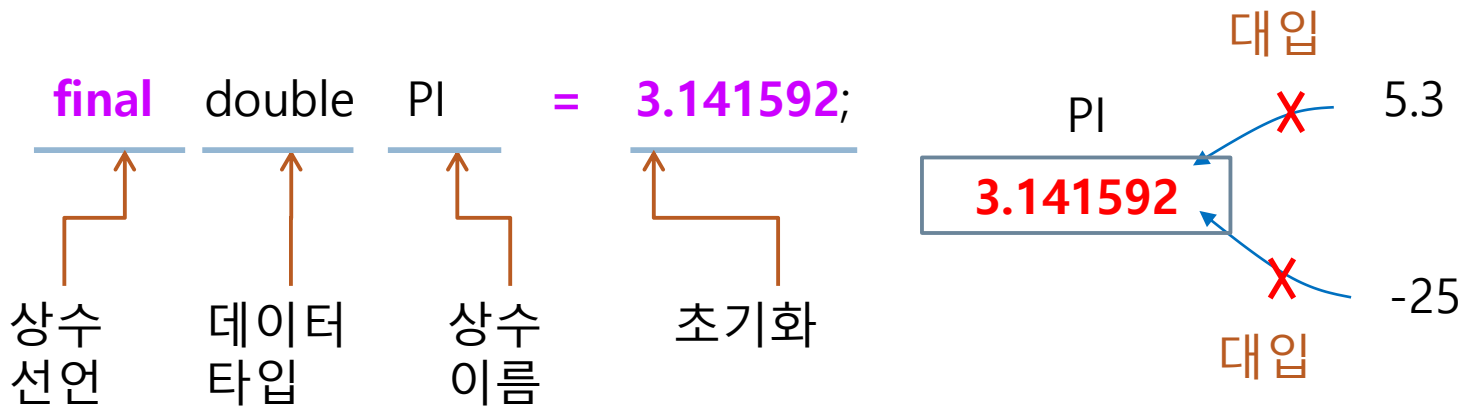
- 허용되지 않는 4가지 경우

```
int x = 15_; // 리터럴 끝에 사용불가
double pi = 3_.14; // 소수점(.) 앞뒤에 사용불가
long idNum = 981231_1234567_L; // _L(_F) 앞에 사용불가
int y = 0_x15; // 0x 의 중간이나 끝에 사용불가 0x_15(오류)
```

상수

22

- 상수 선언
 - ▣ **final** 키워드 사용
 - ▣ 실행 중 값 변경 불가
 - ▣ 선언 시 반드시 초기값 지정



변수, 리터럴, 상수 사용하기

23

원의 면적을 구하는 프로그램을 작성해보자.

```
public class CircleArea {  
    public static void main(String[] args) {  
        final double PI = 3.14; // 원주율을 상수로 선언  
        double radius = 10; // 원의 반지름  
        double circleArea = 0; // 원의 면적  
        circleArea = radius * radius * PI ; // 원의 면적 계산  
  
        // 원의 면적을 화면에 출력한다.  
        System.out.print("원의 면적 = "); // 출력후 줄바꾸지 않음  
        System.out.println(circleArea);  
    }  
}
```

실행 결과

원의 면적 = 314.0

자동 타입 변환

24

- 자동타입 변환이 발생하는 경우
 - ▣ 원래의 타입보다 큰 자료타입으로 바뀔 때

byte >> short/char >> int >> long >> float >> double

- ▣ 원본 데이터 값 그대로 보존 (음수도 그대로 보존)

- 자동 타입 변환 사례

```
byte a;  
int price;  
price = a;
```

정수타입 변수 자동타입 변환 바이트타입 변수

```
long var;  
int n = 32555;  
byte b = 25;  
var = n; // int -> long 자동 변환  
System.out.println("var = (n값) = " + var);  
var = b; // byte -> long 자동변환  
System.out.println("var = (b값) = " + var);
```

실행 결과

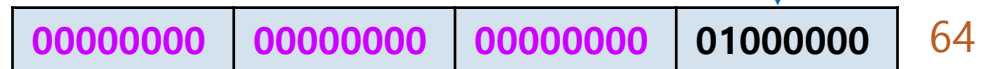
```
var = (n값) = 32555  
var = (b값) = 25
```


자동 타입 변환 사례: byte → int

```
int i, j;  
byte a = 64;  
byte b = -2;  
  
i = a; // 자동타입 변환  
j = b; // 자동타입 변환  
  
System.out.println("i = " + i);  
System.out.println("j = " + j);
```

i = a;

int i

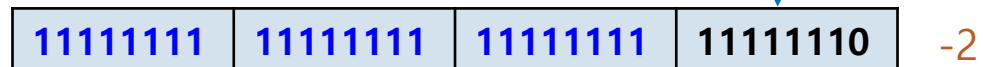


byte a 01000000 64

↓ 변환

j = b;

int j



byte b 11111110 -2


↓ 변환

강제 타입 변환

26

- 강제 타입 변환 :
개발자가 의도적으로 타입 변환
- 강제 타입 변환 방법

```
byte a;  
int price;  
a = (byte) price;
```



- 강제 타입 변환 사례
 - ▣ 실수타입이 정수타입으로 강제 변환 시 **소수점 아래가 버려짐**

```
short var;  
int n = 855638017; // n의 16진수 값은 0x33000001  
var = (short) n;    // int -> short 강제 변환  
System.out.println("var = " + var);  
double d = 2.9;  
n = (int)d;  
System.out.println("n = " + n);
```

실행 결과

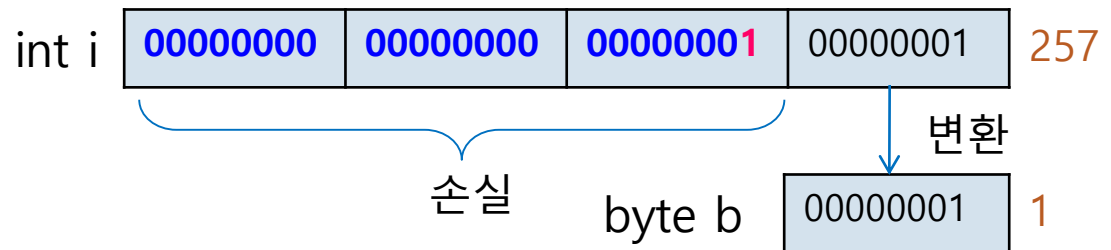
```
var = 1  
n = 2
```

강제 타입 변환 사례 : int → byte , double → int

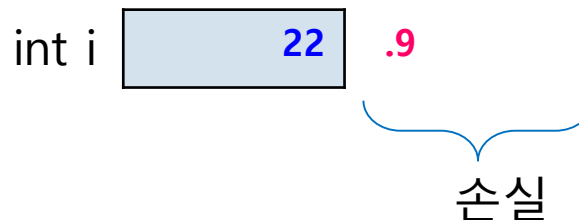
27

```
int i = 257;  
byte b;  
b = (byte)i; // 강제타입 변환  
i = (int)22.9; // 강제타입 변환  
System.out.println("b = " + b);  
System.out.println("i = " + i);
```

b = (byte)i;



i = (int) 22.9;

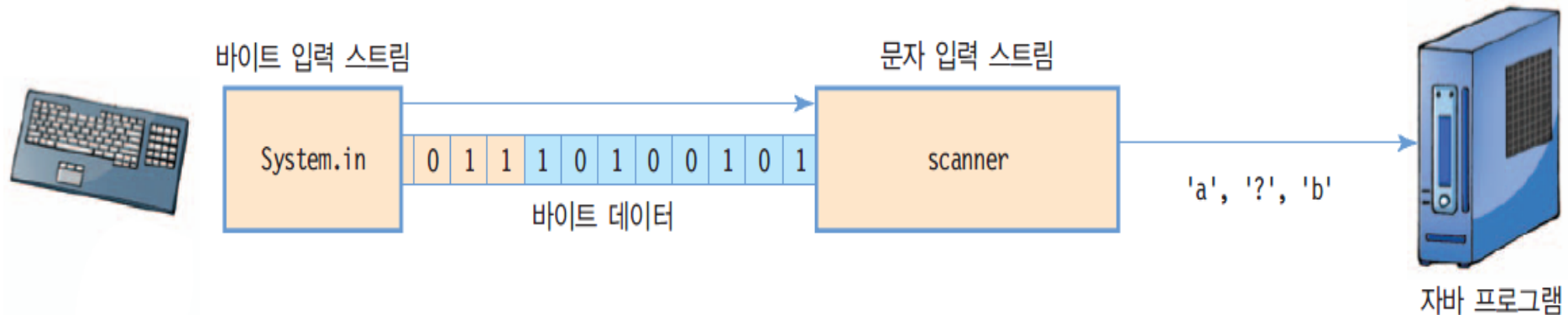


Scanner를 이용한 데이터 입력

28

- Scanner 클래스
 - java.util.Scanner 클래스
 - Scanner 객체 생성

```
Scanner a = new Scanner(System.in);
```



- import문 필요 (소스 맨 앞줄에 씀)

```
import java.util.Scanner;
```

- Scanner에서 데이터 입력 받기
 - Scanner는 입력되는 데이터 값을 공백 ('wt', 'wf', 'Wr', ' ', 'Wn')으로 구분되는 아이템 단위로 읽음

Scanner를 이용한 데이터 입력

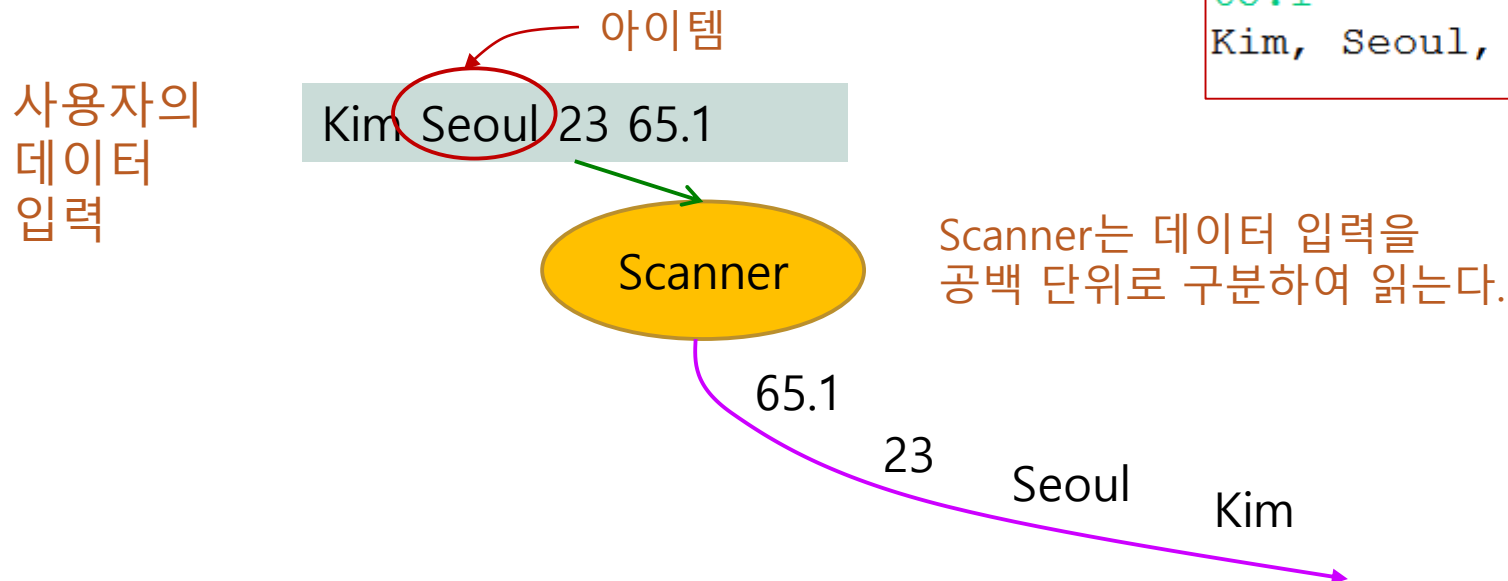
29

```
Scanner scanner = new Scanner(System.in);
```

```
String name = scanner.next();           // "Kim"  
String addr = scanner.next();           // "Seoul"  
int age = scanner.nextInt();             // 23  
double weight = scanner.nextDouble();    // 65.1  
System.out.println(name + ", " + addr + ", "  
    + age + ", " + weight);
```

실행 결과

```
Kim Seoul 23  
65.1  
Kim, Seoul, 23, 65.1
```



Scanner 주요 메소드

30

생성자/메소드	설명
String next()	다음 아이템을 찾아 문자열로 반환
boolean nextBoolean()	다음 아이템을 찾아 boolean으로 변환하여 반환
byte nextByte()	다음 아이템을 찾아 byte로 변환하여 반환
double nextDouble()	다음 아이템을 찾아 double로 변환하여 반환
float nextFloat()	다음 아이템을 찾아 float로 변환하여 반환
int nextInt()	다음 아이템을 찾아 int로 변환하여 반환
long nextLong()	다음 아이템을 찾아 long으로 변환하여 반환
short nextShort()	다음 아이템을 찾아 short로 변환하여 반환
String nextLine()	한 라인 전체('Wn' 포함)를 문자열 타입으로 반환

Scanner를 이용한 데이터 입력 연습

31

Scanner를 이용하여 나이, 체중, 신장 데이터를 키보드에서 입력 받아 다시 출력하는 프로그램을 작성해보자.

```
import java.util.Scanner;
public class ScannerExample {
    public static void main (String args[]) {
        Scanner a = new Scanner(System.in);
        System.out.println("나이, 체중, 신장을 빈칸으로 분리하여 순서대로 입력하세요");
        System.out.println("당신의 나이는 " + a.nextInt() + "살입니다.");
        System.out.println("당신의 체중은 " + a.nextDouble() + "kg입니다.");
        System.out.println("당신의 신장은 " + a.nextDouble() + "cm입니다.");
    }
}
```

데이터 입력부분

나이, 체중, 신장을 빈칸으로 분리하여 순서대로 입력하세요

20 45 165

당신의 나이는 20살입니다.

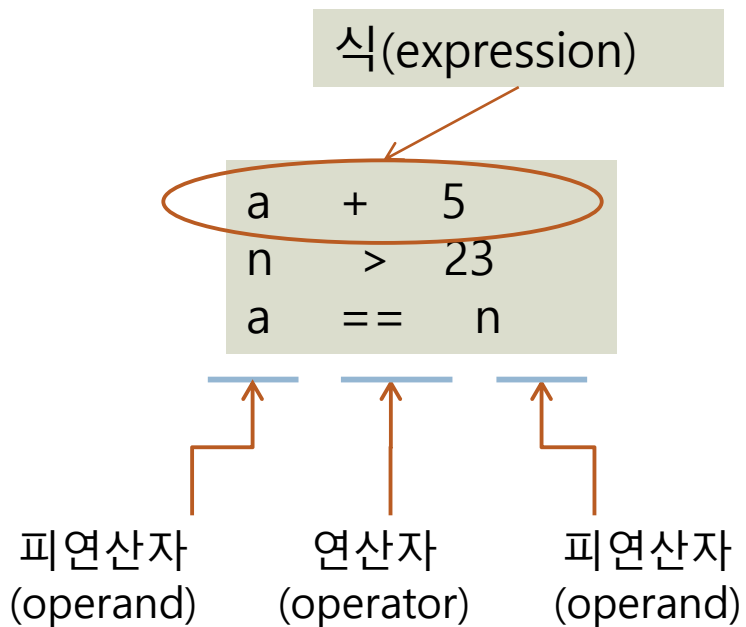
당신의 체중은 45.0kg입니다.

당신의 신장은 165.0cm입니다.

식과 연산자

32

- 연산 : 주어진 식을 계산하여 결과를 얻어내는 과정



연산자의 종류	연산자
증감	++ --
산술	+ - * / %
시프트	>> << >>>
비교	> < >= <= == !=
비트	& ^ ~
논리	&& ! ^
조건	? :
대입	= *= /= += -= &= ^= = <<= >>= >>>=

연산자 우선 순위

33

<div> <div>높음</div> <div>↓</div> <div>낮음</div> </div>	++(postfix) -- (postfix)
	+(양수) -(음수) ++(prefix) --(prefix) ~ !
	형 변환(type casting)
	* / %
	+(덧셈) -(뺄셈)
	<< >> >>>
	< > <= >= instanceof
	== !=
	& (비트 AND)
	^ (비트 XOR)
	(비트 OR)
	&& (논리 AND)
	(논리 OR)
	? : (조건)
	= += -= *= /= %= &= ^= = <<= >>= >>>=

- 같은 우선순위의 연산자
 - ▣ 왼쪽에서 오른쪽으로 처리
 - ▣ 예외) 오른쪽에서 왼쪽으로
 - 대입 연산자, --, ++, +,-(양수 음수 부호), !, 형 변환은 오른쪽에서 왼쪽으로 처리
- 괄호는 최우선순위
 - ▣ 괄호가 다시 괄호를 포함한 경우는 가장 안쪽의 괄호부터 먼저 처리

산술 연산자

34

산술 연산자	의미	예	결과값
+	더하기	$25.5 + 3.6$	29.1
-	빼기	$3 - 5$	-2
*	곱하기	$2.5 * 4$	10.0
/	나누기	$5/2$	2
%	나머지	$5\%2$	1

산술 연산 예제

35

500초는 몇 시간, 몇 분, 몇 초인가?

```
import java.util.Scanner;
public class ArithmeticOperator {
    public static void main (String[] args) {
        int time, second, minute, hour;
        Scanner sc = new Scanner(System.in);
        System.out.print("정수를 입력하세요:"); // 시, 분, 초로 변환될 정수 입력

        time = sc.nextInt();
        second = time % 60; // 초
        minute = (time / 60) % 60; // 분
        hour = (time / 60) / 60; // 시간

        System.out.print(time + "초는 ");
        System.out.print(hour + "시간, ");
        System.out.print(minute + "분, ");
        System.out.println(second + "초입니다.");
    }
}
```

정수를 입력하세요:5000

5000초는 1시간, 23분, 20초입니다.

비트 연산자

36

- 피 연산자의 각 비트들을 대상으로 하는 연산

비트 연산자	내용
$a \& b$	a와 b의 각 비트들의 AND 연산. 두 비트 모두 1일 때만 1이 되며 나머지는 0
$a b$	a와 b의 각 비트들의 OR 연산. 두 비트 모두 0일 때만 0이 되며 나머지는 1
$a \wedge b$	a와 b의 각 비트들의 XOR 연산. 두 비트가 서로 다르면 1, 같으면 0
$\sim a$	단항 연산자로서, a의 각 비트들에 NOT 연산. 1을 0으로, 0을 1로 변환

비트 연산자의 사례

37

$$\begin{array}{r} 01101010 \\ \& 11001101 \\ \hline 01001000 \end{array}$$

모두 1이므로
결과는 1

둘 중 하나라도 0이
되면 결과는 0

$$\begin{array}{r} 01101010 \\ | 11001101 \\ \hline 11101111 \end{array}$$

모두 0이므로
결과는 0

둘 중 하나라도 1이
되면 결과는 1

$$\begin{array}{r} 01101010 \\ \wedge 11001101 \\ \hline 10100111 \end{array}$$

두 비트가 모두
같으므로
결과는 0

두 비트가 서로
다르므로 결과는 1

$$\begin{array}{r} \sim 01101010 \\ \hline 10010101 \end{array}$$

0은 1로
변환

1은 0으로
변환

시프트 연산자

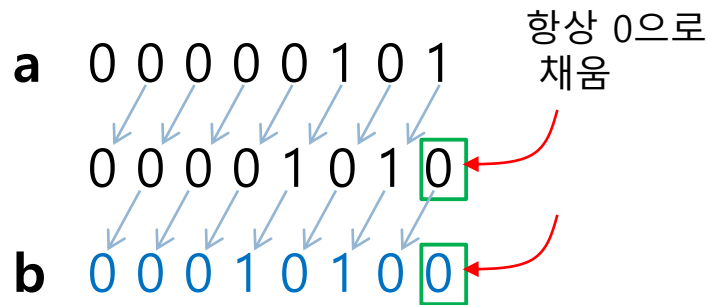
38

시프트 연산자	내용
$a \gg b$	<p>a의 각 비트를 오른쪽으로 b 번 시프트한다.</p> <p>최상위 비트의 빈자리는 시프트 전의 최상위 비트로 다시 채운다.</p> <p>산술적 오른쪽 시프트.</p>
$a >>> b$	<p>a의 각 비트를 오른쪽으로 b 번 시프트한다.</p> <p>그리고 최상위 비트의 빈자리는 0으로 채운다.</p> <p>논리적 오른쪽 시프트.</p>
$a \ll b$	<p>a의 각 비트를 왼쪽으로 b 번 시프트한다.</p> <p>그리고 최하위 비트의 빈자리는 0으로 채운다.</p> <p>산술적 왼쪽 시프트.</p>

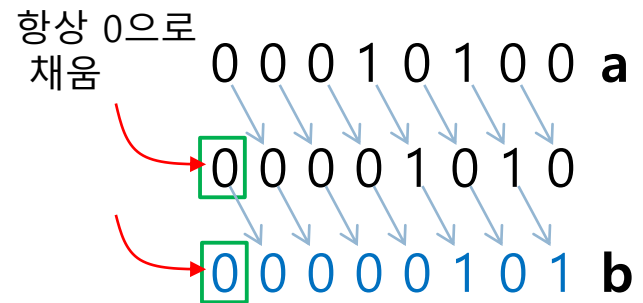
시프트 연산자의 사례 (1/2)

39

```
byte a = 5; // 5  
byte b = (byte)(a << 2); // 20
```



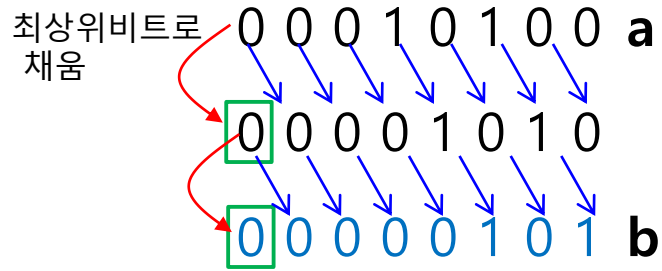
```
byte a = 20; // 20  
byte b = (byte)(a >>> 2); // 5
```



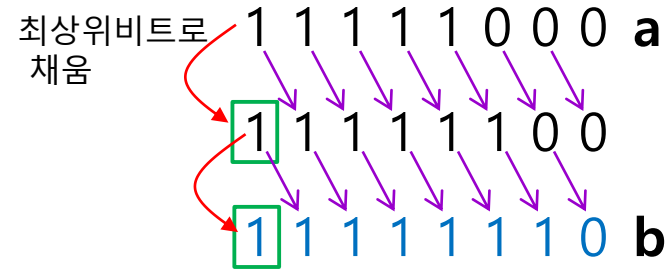
시프트 연산자의 사례 (2/2)

40

```
byte a = 20; // 20  
byte b = (byte)(a >> 2); // 5
```



```
byte a = (byte)0xf8; // -8  
byte b = (byte)(a >> 2); // -2
```



Tip: 산술적 시프트와 논리적 시프트

41

- 산술적 오른쪽 시프트
 - ▣ >>는 1비트 오른쪽으로 시프트할 때마다 2로 나누기하는 결과
- 산술적 왼쪽 시프트
 - ▣ << 연산자는 1비트 시프트할 때마다 2로 곱하는 결과
 - ▣ 음수(최상위 비트가 1)는 시프트 결과 최상위 비트가 0인 양수가 되는 오버플로 발생 가능 주의
- 논리적 오른쪽 시프트
 - ▣ >>>는 시프트 시 최상위 비트에 항상 0이 삽입
 - ▣ 나누기의 산술적 효과가 나타나지 않음
- byte, short, char 타입의 시프트 연산 시 주의 사항
 - ▣ int 타입으로 변환되어 연산이 일어나므로 원하지 않는 결과 발생 가능

비교연산자

42

비교 연산자	내용	예제	결과
$a < b$	a가 b보다 작으면 true 아니면 false	$3 < 5$	true
$a > b$	a가 b보다 크면 true 아니면 false	$3 > 5$	false
$a \leq b$	a가 b보다 작거나 같으면 true 아니면 false	$1 \leq 0$	false
$a \geq b$	a가 b보다 크거나 같으면 true 아니면 false	$10 \geq 10$	true
$a == b$	a가 b와 같으면 true 아니면 false	$1 == 3$	false
$a != b$	a가 b와 같지 않으면 true 아니면 false	$1 != 3$	true

논리 연산자 1

43

a	!a	예제
true	false	!(3 < 5)는 false
false	true	!(3 > 5)는 true

a	b	a ^ b	예제
true	true	false	(3<5) ^ (1==1)은 false
true	false	true	(3<5) ^ (1==2)은 true
false	true	true	(3>5) ^ (1==1)은 true
false	false	false	(3>5) ^ (1==2)은 false

논리 연산자 2

44

a	b	a b	예제
true	true	true	(3<5) (1==1)은 true
true	false	true	(3<5) (1==2)은 true
false	true	true	(3>5) (1==1)은 true
false	false	false	(3>5) (1==2)은 false

a	b	a && b	예제
true	true	true	(3<5) && (1==1)은 true
true	false	false	(3<5) && (1==2)은 false
false	true	false	(3>5) && (1==1)은 false
false	false	false	(3>5) && (1==2)은 false

대입 연산자, 증감 연산자

45

대입 연산자	내용
<code>a = b</code>	b의 값을 a에 대입
<code>a += b</code>	<code>a = a + b</code> 과 동일
<code>a -= b</code>	<code>a = a - b</code> 과 동일
<code>a *= b</code>	<code>a = a * b</code> 과 동일
<code>a /= b</code>	<code>a = a / b</code> 과 동일
<code>a %= b</code>	<code>a = a % b</code> 과 동일
<code>a &= b</code>	<code>a = a & b</code> 과 동일
<code>a ^= b</code>	<code>a = a ^ b</code> 과 동일
<code>a = b</code>	<code>a = a b</code> 과 동일
<code>a <<= b</code>	<code>a = a << b</code> 과 동일
<code>a >>= b</code>	<code>a = a >> b</code> 과 동일
<code>a >>>= b</code>	<code>a = a >>> b</code> 과 동일

증감연산자	내용
<code>a++</code>	a을 먼저 사용한 후에 1 증가
<code>a--</code>	a을 먼저 사용한 후에 1 감소
<code>++a</code>	a을 먼저 1 증가한 후에 사용
<code>--a</code>	a을 먼저 1 감소한 후에 사용

증감 연산자

46

- 증감 연산의 순서
 - ▣ 연산자가 피연산자 뒤에 붙는 경우

```
int a, b = 4;  
a = b++;  
// 결과 a=4, b=5
```

- ▣ 연산자가 피연산자 앞에 붙는 경우

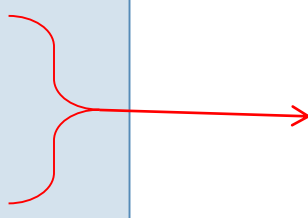
```
int a, b = 4;  
a = ++b;  
// 결과 a=5, b=5
```

조건 연산자 ? :

47

- `opr1 ? opr2 : opr3`
 - ▣ 세 개의 피연산자로 구성되어 삼항(ternary) 연산자
 - ▣ `opr1`이 true이면 값은 `opr2`, false이면 `opr3`.
 - ▣ if-else에 비행 문장이 간결해짐

```
int x = 5;  
int y = 3;  
int s;  
if (x > y)  
    s = 1;  
else  
    s = -1;
```



```
int s = (x > y)? 1 : -1 ;
```

조건 연산자 사용하기

48

다음 소스의 실행 결과는 무엇인가?

```
public class TernaryOperator {  
    public static void main (String[] args) {  
        int a = 3, b = 5;  
        System.out.println("두 수의 차는 " + ((a>b)?(a-b):(b-a)));  
    }  
}
```

실행결과

두 수의 차는 2

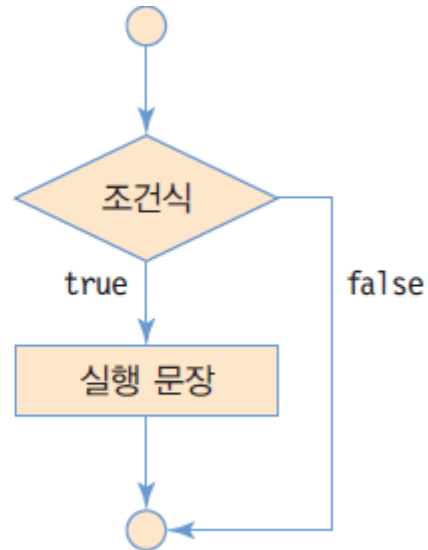
조건문 – if문

49

- 단순 if 문
 - ▣ if 다음의 괄호 안에는 조건식(논리형 변수나 논리 연산)
 - ▣ 조건식의 값
 - true인 경우, if문을 벗어나 다음 문장이 실행된다.
 - false의 경우에는 if 다음의 문장이 실행되지 않고 if 문을 빠져 나온다.

```
if(조건식) {  
    ...실행 문장...  
}
```

if 키워드



if문 사용하기

50

시험 점수가 80점이 이상이면 합격 판별을 하는 프로그램을 작성하시오.

```
import java.util.Scanner;
public class SuccessOrFail {
    public static void main (String[] args) {
        Scanner in = new Scanner(System.in);

        System.out.print("점수를 입력하시오: ");
        int score = in.nextInt();
        if (score >= 80)
            System.out.println("축하합니다! 합격입니다.");
    }
}
```

실행결과

점수를 입력하시오: 95
축하합니다! 합격입니다.

조건문 – if-else

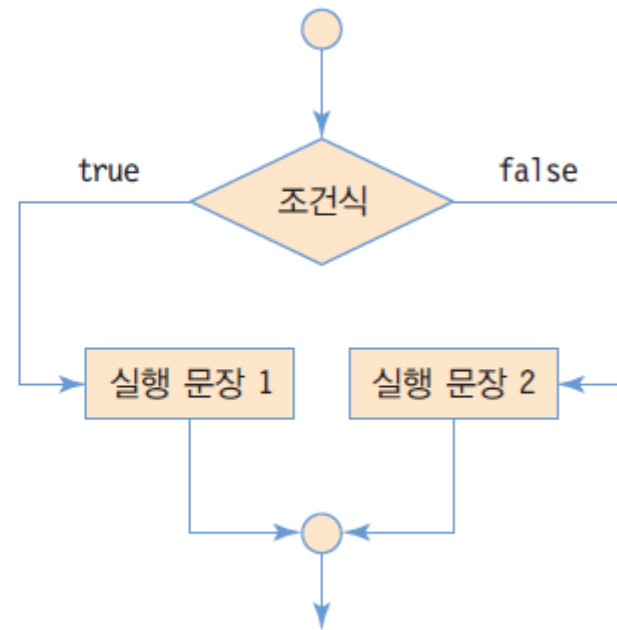
51

- if-else 문
 - ▣ 조건식이 true면 실행문장1 실행 후 if-else문을 벗어남
 - ▣ false인 경우에 실행문장2 실행후, if-else문을 벗어남

```
if(조건식) {  
    ...실행 문장 1...  
}  
else {  
    ...실행 문장 2...  
}
```

if 키워드

else 키워드



if-else 사용하기

52

입력된 수가 3의 배수인지 판별하는 프로그램을 작성하시오.

```
import java.util.Scanner;
public class MultipleOfThree {
    public static void main (String[] args) {
        Scanner in = new Scanner(System.in);

        System.out.print("수를 입력하시오: ");
        int number = in.nextInt();

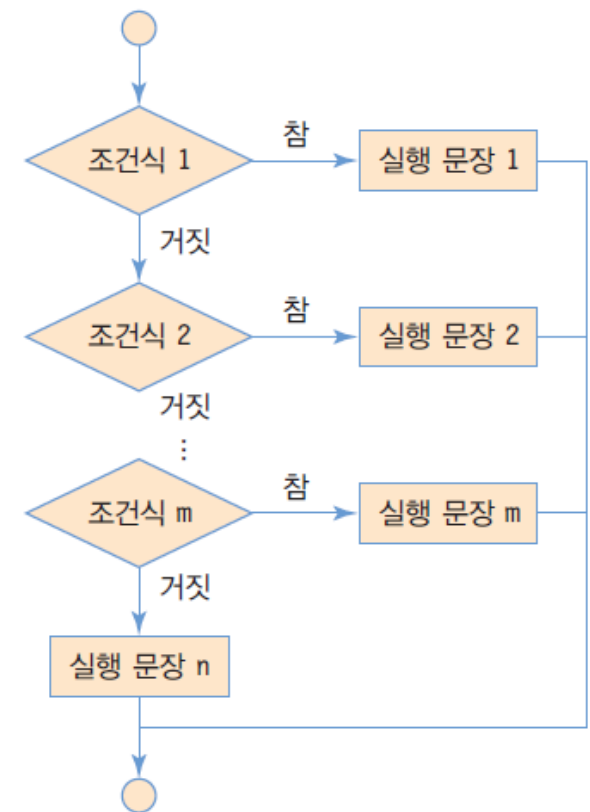
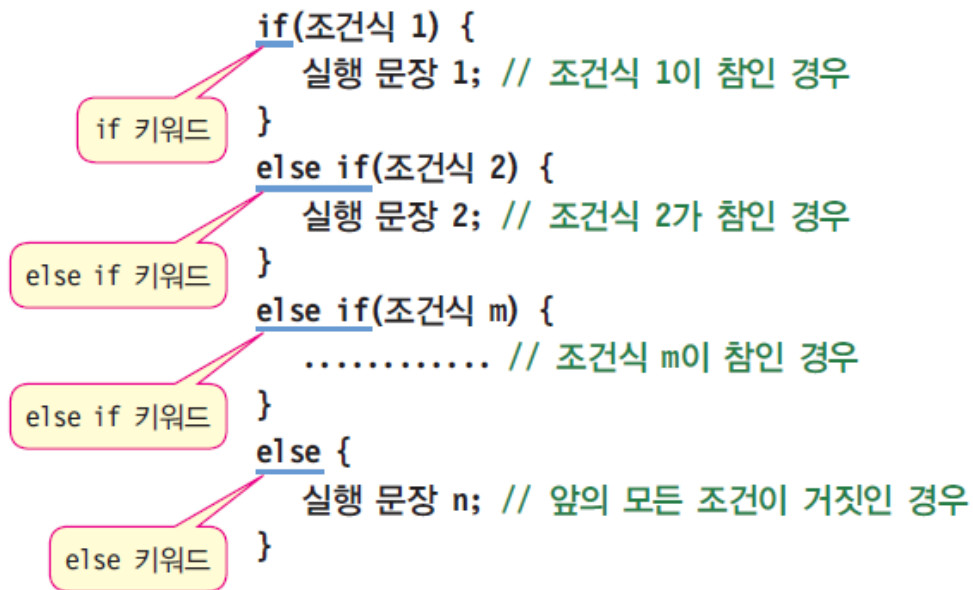
        if (number % 3 == 0)
            System.out.println("3의 배수입니다.");
        else
            System.out.println("3의 배수가 아닙니다.");
    }
}
```

수를 입력하시오: 129
3의 배수입니다.

조건문 - 다중 if

53

- 다중 if문
 - ▣ 실행 문장이 다시 if문 또는 if-else문을 포함하는 경우를 말함
 - ▣ Else 문은 바로 전의 if문과 짝을 이룬다.
 - ▣ 조건문이 너무 많은 경우, switch 문 사용 권장



학점 매기기

54

if-else문을 이용하여 키보드 입력된 성적에 대해 학점을 부여하는 프로그램을 작성해보자.

```
import java.util.Scanner;
public class Grading {
    public static void main (String[] args) {
        char grade;
        Scanner a = new Scanner(System.in);
        while (a.hasNext()) {
            int score = a.nextInt();
            if(score >= 90.0)
                grade = 'A';
            else if(score >= 80.0)
                grade = 'B';
            else if(score >= 70.0)
                grade = 'C';
            else if(score >= 60.0)
                grade = 'D';
            else
                grade = 'F';
            System.out.println("학점은 "+grade+"입니다");
        }
    }
}
```

키가 입력될 때까지 기다리며, 입력된 키가 있는 경우 true 리턴. ctrl-z 키가 입력되면 false 리턴

실행결과

80
학점은 B입니다
90
학점은 A입니다
76
학점은 C입니다

switch문

55

- switch문은 식과 case 문의 값과 비교
 - ▣ case의 비교 값과 일치하면 해당 case문의 실행 문장 수행.
 - break를 만나면 switch문을 벗어남
 - ▣ case의 비교 값과 일치하는 것이 없으면 default 문 실행
- default문은 생략 가능

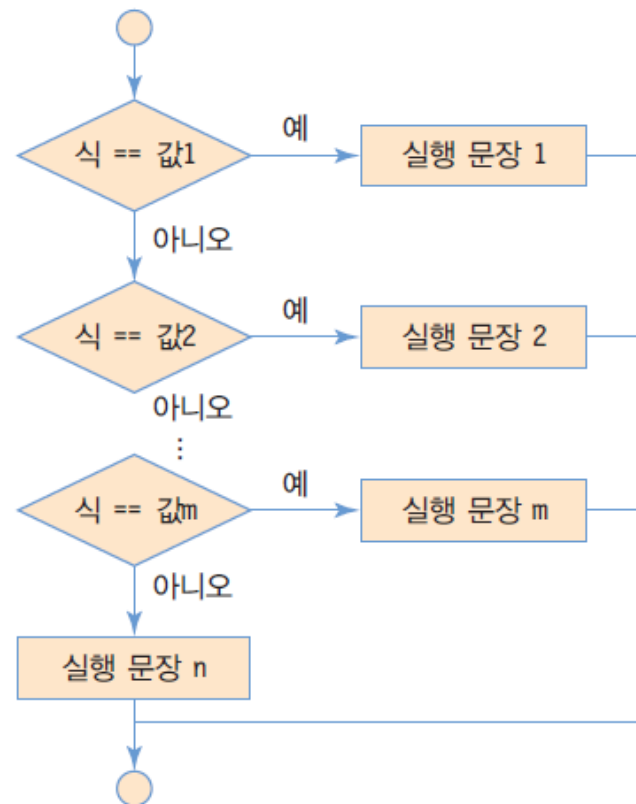
```
switch(식) {  
    case 값1: 실행 문장 1;  
        break;  
    case 값2: 실행 문장 2;  
        break;  
    ...  
    case 값m: 실행 문장 m;  
        break;  
    default: 실행 문장 n;  
}
```

switch 키워드

case 키워드

break 키워드

default 키워드



switch문에서 벗어나기

56

- switch문 내의 break문
 - ▣ break 문장을 만나면 switch문을 벗어남
 - ▣ 만일 case 문에 break문이 없다면, 다음 case문의 실행 문장으로 실행을 계속하게 되며, 언젠가 break를 만날 때까지 계속 내려감

```
import java.util.Scanner;
public class Grading {
    public static void main (String[] args) {
        char grade='A';
        switch (grade) {
            case 'A':
                System.out.println("90~100점입니다.");
                break;
            case 'B':
                System.out.println("80~89점입니다.");
                break;
            case 'C':
                System.out.println("70~79점입니다.");
                break;
        }
    }
}
```


switch문의 break 사용하기

57

학점이 A, B 인 학생에게는 "참 잘하였습니다.", 학점이 C, D인 학생에게는 "좀 더 노력하세요.", 학점이 F인 학생에게는 "다음 학기에 다시 수강하세요."를 출력하는 프로그램을 switch문의 break를 잘 활용하여 작성하시오.

```
public class GradeSwitch {  
    public static void main(String[] args) {  
        char grade='C';  
        switch (grade) {  
            case 'A':  
            case 'B':  
                System.out.println("참 잘하였습니다.");  
                break;  
            case 'C':  
            case 'D':  
                System.out.println("좀 더 노력하세요.");  
                break;  
            case 'F':  
                System.out.println("다음 학기에 다시 수강하세요.");  
                break;  
            default:  
                System.out.println("잘못된 학점입니다.");  
        }  
    }  
}
```

실행결과 좀 더 노력하세요.

case 문의 값

58

□ case 문의 값의 특징

- switch 문은 식의 결과 값을 case 문과 비교
- 사용 가능한 case문의 비교 값
 - 정수 타입 리터럴, JDK 1.7부터는 문자열 리터럴도 허용

```
public class GradeSwitch {  
    public static void main(String[] args) {  
        int a = 0;  
        int b = 1;  
        int c = 25;  
        switch(c%2) {  
            case 1 : // 정수 리터럴  
                //...;  
                break;  
            case 2: // 정수 리터럴  
                // ...;  
                break;  
        }  
        String s = "예";  
        switch(s) {  
            case "예" : // 문자열 리터럴 사용 가능. JDK1.7부터 적용  
                //...;  
                break;  
            case "아니요" : // 문자열 리터럴 사용 가능. JDK1.7부터 적용  
                //...;  
                break;  
        }  
    }  
}
```

```
switch(a) {  
    case a :           // 오류. 변수 사용 안됨  
    case a > 3 :       // 오류. 수식 안됨  
    case a == 1 :     // 오류. 수식 안됨  
}
```

잘못된 case 문

정상적인 case 문

성적 분류

59

앞의 다중 if문을
이용한 성적 분류
프로그램을 switch
문으로 바꾸시오.

실행결과

100
학점은 A입니다
55
학점은 F입니다
76
학점은 C입니다

```
import java.util.Scanner;
public class Grading2 {
    public static void main (String[] args) {
        char grade;
        Scanner a = new Scanner(System.in);
        while (a.hasNext()) {
            int score = a.nextInt();
            switch (score/10) {
                case 10:
                case 9:
                    grade = 'A';
                    break;
                case 8:
                    grade = 'B';
                    break;
                case 7:
                    grade = 'C';
                    break;
                case 6:
                    grade = 'D';
                    break;
                default:
                    grade = 'F';
            }
            System.out.println("학점은 "+grade+"입니다");
        }
    }
}
```