JAVA 기초 강의

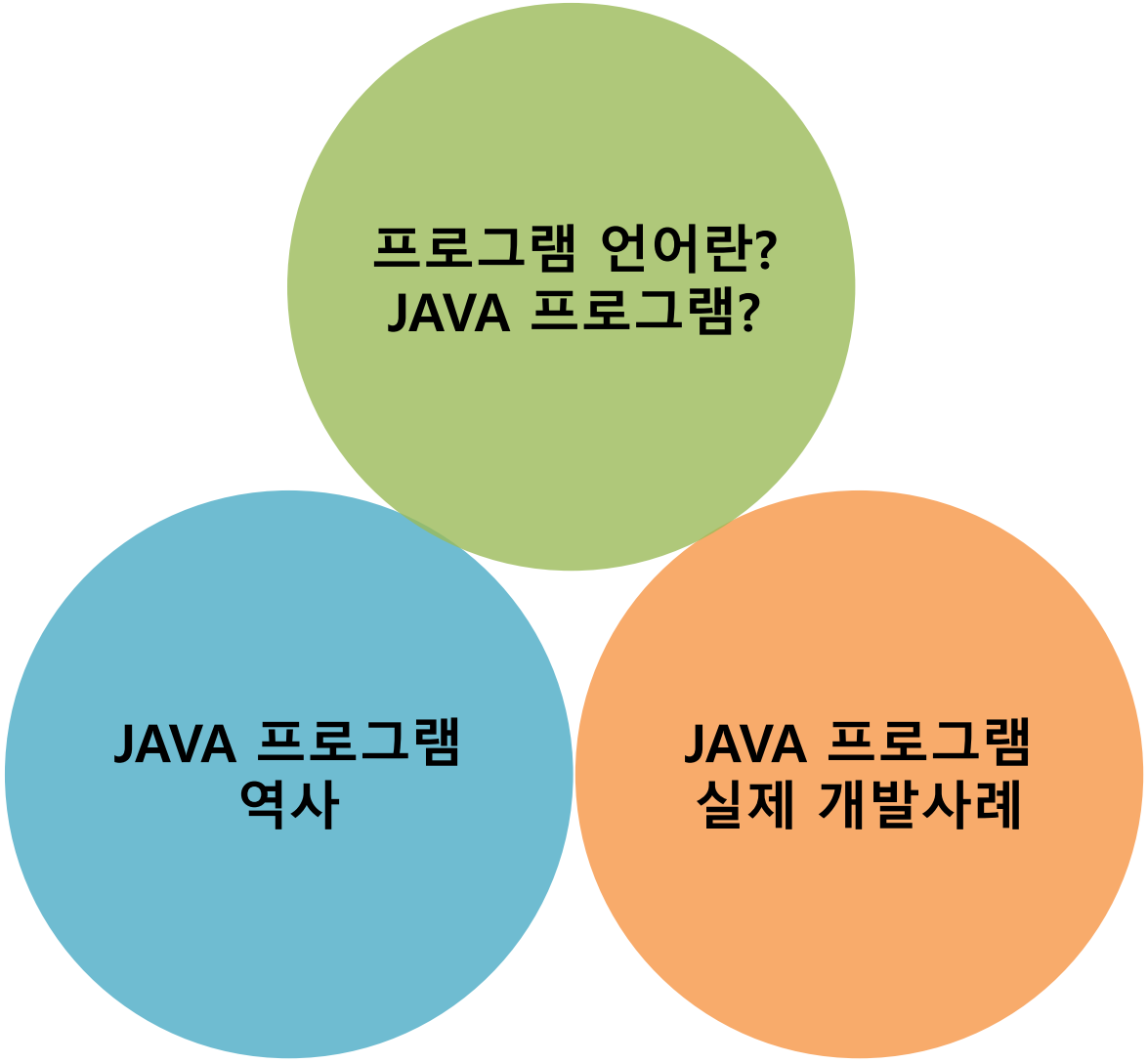
2022. 2

강 현 준

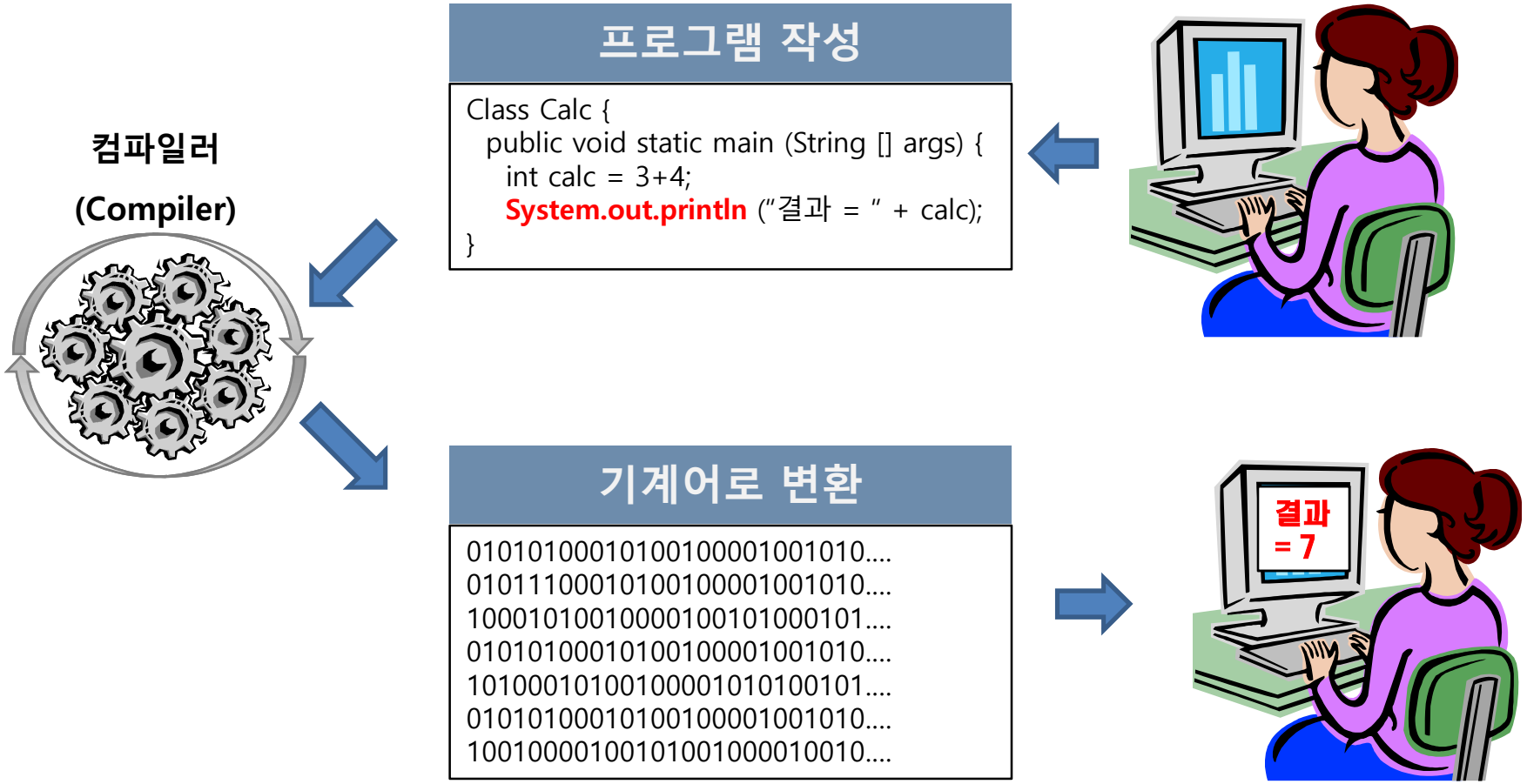
human@human.or.kr

1장. JAVA 시작하기

1.1 자바 개요



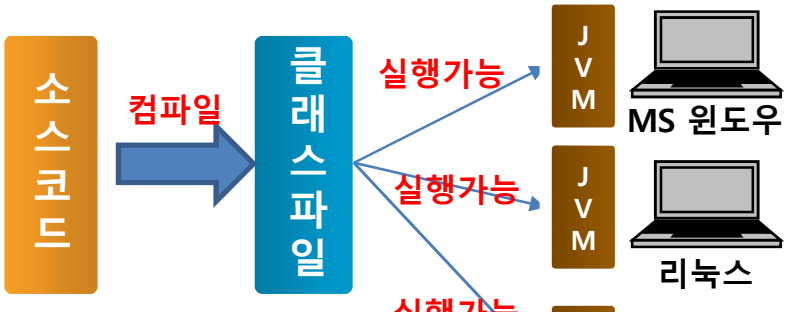
1.1.1. 프로그램 언어란?


프로그램 작성
Class Calc {
public void static main (String [] args) {
int calc = 3+4;
System.out.println ("결과 = " + calc);
}
컴파일러
(Compiler)
기계어로 변환
01010100010100100001001010....
01011100010100100001001010....
10001010010000100101000101....
01010100010100100001001010....
10100010100100001010100101....
01010100010100100001001010....
10010000100101001000010010....
결과
= 7

1.1.2. JAVA 프로그램의 특징?

특징 1

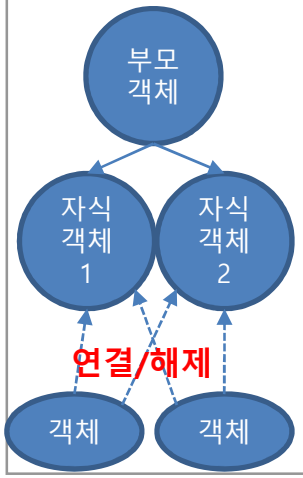
이식성이 높은 언어 (운영체제에 독립)



* 클래스 파일 : 기계어로 해석된 파일을 의미함
* JVM (Java Virtual Machine)
 - 자바가상머신 (운영체제별로 별도 설치)

특징 2

객체지향 언어



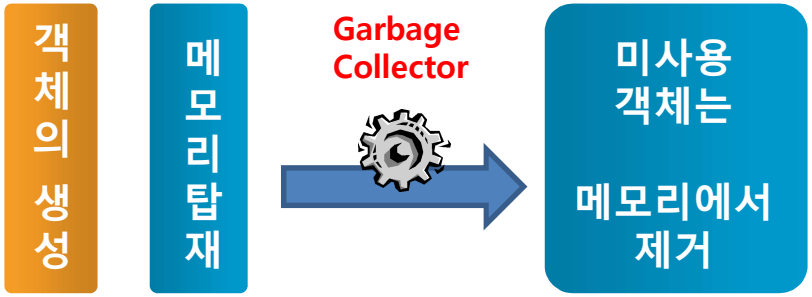
1. 자연적인 모델링 :
실제 사물을 표현하듯이 하나의 객체로 생각하면서 프로그램이 이루어짐.
일상 생활의 모습과 대비하면서 자연스럽게 구현가능.

2. 재활용 및 생산성 향상
상속을 통해서 코드의 재활용성을 고려가능.

3. 유지보수 우수성
잘 설계된 객체는 모듈끼리의 의존성이 떨어집니다. 객체의 교체를 통해 프로그램의 추가/수정이 용이함.

특징 3

메모리 자동관리



* C++는 메모리를 반드시 해제해주어야 하는 코드를 개발자가 직접 성실하게 작성해야 함.

1.1.2. JAVA 프로그램의 특징?

특징 4

다양한 애플리케이션 개발 가능

명령어 기반의 Console / 사용자 화면 기반의 UI 프로그램
인터넷 기반의 서버 프로그램 / 안드로이드 모바일 프로그램

[JAVA 개발도구 및 API]

JAVA SE
(Standard Edition)
- JAVA 프로그램 최소한의 JVM 제공
- 프로그램 개발도구와 API 정의
- JDK 형태로 제공함. (각자 PC 설치 가능)

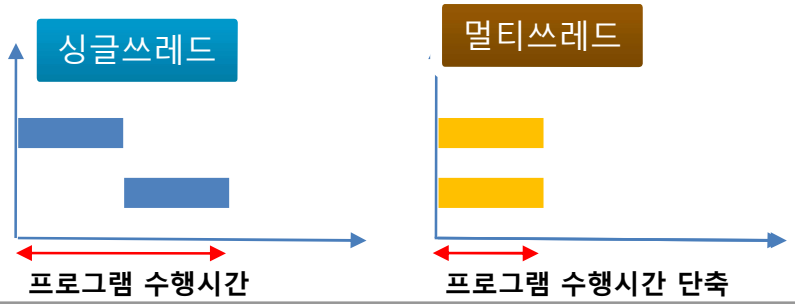
JAVA EE
(Enterprise Edition)
- 서버 애플리케이션 개발시 사용함.
- WAS(Web Application Server)의 형태로 제공 (Tomcat, WebLogic 등)

JAVA ME
(Micro Edition)
- 임베디드 / 모바일 장비를 위한 개발도구 및 API 제공

특징 5

멀티 쓰레드 지원

동시에 여러 작업을 할 경우 / 대용량 작업을 처리할 경우
시스템의 성능 개선을 위해 병렬처리가 필요함.
멀티 쓰레드 기능으로 지원 가능.



특징 6

막강 Open 라이브러리 풍부

- JAVA는 Open 소스 기반이므로
  Java로 개발된 라이브러리도 많음.

- 기 개발된 라이브러리 활용시
  프로그램 개발에 들어가는 노력과 비용을 줄일 수 있고,
  유지보수의 편의성을 위해서 사용이 가능함.

1.1.3. JAVA 프로그램의 역사

➢ JAVA는 1996년부터 2~3년마다 새로운 버전이 발표됨. (안정성 때문)
➢ 그러나 JAVA SE 9 이후부터는 변화에 맞추기 위해 6개월마다 버전 발표함.

- JDK 1.0 (1996. 01. 23)
- JDK 1.1 (1996. 02. 19)
- J2SE 1.2 (1998. 12. 08) :
- J2SE 1.3 (2000. 05. 08)
- J2SE 1.4 (2002. 02. 06)
- J2SE 5.0 (2004. 09. 30) :
- Java SE 6 (2006. 12. 11)
- Java SE 7 (2011. 07. 28)
- Java SE 8 (2014. 03. 18) :
- Java SE 9 (2017. 09. 21)
- Java SE 10 (2018. 03. 20)
- Java SE 11 (2018. 09. 25)
- Java SE 12 (2019. 03. 19)
- .. 현재도 지속 발표되고 있음 (현재기준 18Ver)

- 이때부터 JAVA 2로 명명함. 이 때 JAVA의 많은 형태가 확정됨.
- J2SE, J2EE, J2ME 의 형태로 완성됨.

- JAVA 1.5라고도 함. 이 때 JAVA의 많은 기능이 추가됨.

- 이 때도 JAVA의 많은 기능이 추가됨.

JDK : Java Development Kit
(JAVA 개발 도구)

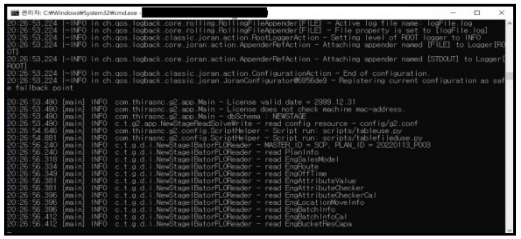
1.1.4. JAVA 실제 구현사례

PC 어플리케이션

PC에서 구동되는 프로그램

JAVA 개발도구

생산스케줄 프로그램



웹 애플리케이션

웹 서버를 이용한 웹사이트 프로그램

JAVA 개발 Framework

생산계획 솔루션

모바일 앱

안드로이드 프로그램

모바일 개발 Tool

모바일 솔루션

기타

빅데이터

게임

1.2.1. JAVA 개발 환경 구축

JDK 및 JRE 다운로드

1. JDK와 JRE의 차이점
   - JDK : JVM + 라이브러리 + 컴파일러
   - JRE : JVM + 라이브러리

2. JAVA를 통해 개발하려면 JDK 설치하면 되고,
   JAVA로 개발된 파일을 통해서 구동만 하려면 JRE만 설치하면 됨

3. JDK는 오라클 웹사이트에서 다운로드 가능함. (www.oracle.com)

환경설정

1. 컴파일러 : javac.exe
2. JAVA 구동 : java.exe
3. Path 설정 : java설치파일의 bin 폴더 path 설정

이클립스설치

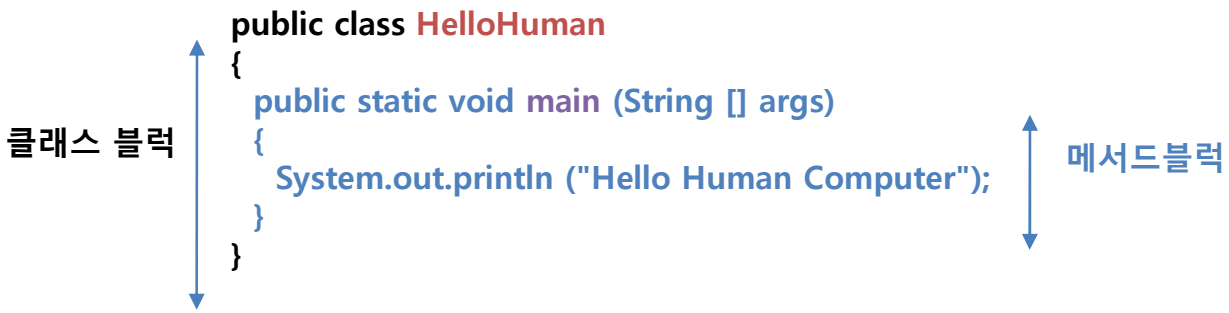
1. Java 개발도구.
2. www.eclipse.org에서 무료 사용가능
3. 워크스페이스 설정 : 프로그램을 개발하여 관리할 주요 위치
4. 퍼스펙티브 및 뷰 설정

1.2.2. JAVA 소스분석

1. 클래스 : 필드 또는 메서드를 포함하는 블록
- **아래 소스 중 HelloHuman이 클래스임.**
- **주의사항 :**
  - ▷ 클래스명과 파일명이 일치해야 함.
  - ▷ 첫글자는 대문자
  - ▷ 숫자로 시작하면 안됨. (ex. 203HelloHuman은 안됨.)
  - ▷ 공백을 포함해서도 안됨. 필요시 "_"를 붙이도록 할 것
  - ▷ 중괄호 "{ ... }"로 시작과 끝을 안내함.

2. 메서드 : 어떤일을 처리하는 실행문을 포함하는 블록
- **아래 소스 중 main이 메서드를 의미함.**
- **주의사항 :**
  - ▷ 개발자가 마음대로 정의가능. (단, Main 메서드는 변형불가)
  - ▷ 중괄호 "{ ... }"로 시작과 끝을 안내함.

[파일명 HelloHuman.java]



1.2.3. 주석

3. 주석

- **실행되지 않는 코드문**
- **코드에 설명을 붙일 때 주로 사용함.**
- **형태**

주석기호	설명
/* .... */	/* ~ */ 사이에 있는 모든 문장을 주석으로 인지함
//	// 가 있는 부분부터 라인 끝까지를 주석으로 인지함.

- ex1 : /* ... */ 주석 예제

```
public class HelloHuman
{
  public static void main (String [] args)
  {
     System.out.println ("Hello Human Computer1");
/*   2022.5.8 주석 추가하여 임시 TEST
     System.out.println ("Hello Human Computer2");
     System.out.println ("Hello Human Computer3");
*/
  }
}
```

[출력예제]

Hello Human Computer1

*** 주석 처리된**
Hello Human Computer2
Hello Human Computer3
은 출력안됨.

- ex2 : // 주석 예제

```
public class HelloHuman
{
  public static void main (String [] args)
  {
     System.out.println ("Hello Human Computer1");
//   System.out.println ("Hello Human Computer2");
     System.out.println ("Hello Human Computer3");
  }
}
```

[출력예제]

Hello Human Computer1
Hello Human Computer3

*** Hello Human Computer2는**
출력안됨.

1.2.4. 실행문

4. 실행문

- 실행문은 변수 선언, 값 저장, 메소드 호출 등에 해당하는 코드
- 실행문의 마지막에는 세미콜론(;)을 붙여서 문장의 끝을 알려야 함.
- 한 줄에 여려개의 문장끝을 알려도 문제없음
- 아래의 2개 예제는 같은 결과임. (단, 가독성을 위해서 ex1을 권장함.)

- ex1 :

```
public class HelloHuman
{
  public static void main (String [] args)
  {
    int kor = 100;
    int eng = 90;
    int math = 95;
    System.out.println (kor+eng+math);
  }
}
```

[출력예제]

285

- ex2 :

```
public class HelloHuman
{
  public static void main (String [] args)
  {
    int kor = 100; int eng = 90; int math = 95;
    System.out.println (kor+eng+math);
  }
}
```

[출력예제]

285

2장. 변수와 타입

2.1. 변수

- 변수란 하나의 값을 저장하는 메모리 공간이다.
- 변수를 사용하기 위해서는 변수를 선언해야 함.

변수 상세 설명

선언

1. 변수의 선언
- 타입 + 변수명 ;

2. 선언 예시

```
int age;      // 정수값을 저장할 수 있는 age 변수의 선언
double average; // 실수값을 저장할 수 있는 avg변수의 선언
int x, y, z;   // x,y,z의 변수를 정수형으로 선언
```

자바예약어

분류	예 시
기본타입	boolean, byte, int, long, float, double 등
접근지정자	private, protected, public 등
클래스 관련	class, abstract, interface, extends, implements 등
객체 관련	new, instanceof, this, super, null 등
메소드 관련	void, return 등
제어문 관련	if, else, switch, break, continue, for, while 등
기타	try, catch, finally, throw, true, false 등

변수명 작성규칙	예 시
첫번째글자는 문자, $, _ 이어야 하고, 숫자가 될 수 없음	가능 : human, $human, _human 불가 : @human, $#human
영어 대/소문자 구분함	Human과 human은 서로 다른 문자임
관례적으로 첫문자는 소문자로 시작하되, 다른 단어와 혼용할 경우 대문자로 구분	예시) maxPrice, minSpeed 등
자바 예약어는 사용 불가함	위의 표 참조
문자수는 제한없음	

2.1. 변수

- 변수를 사용하는 것은 변수에 값을 저장 및 활용하는 행위를 의미함.

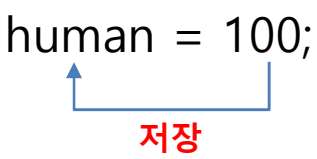
변수 사용 예제

변수저장

```
int human;        // 변수의 선언
human = 100;   // human에 100이란 정수를 저장

또는

int human = 100 ;        // 위와 같은 문장임.
```



변수활용

```
System.out.print(human);              // 화면에 human에 담겨진 100 이란 숫자가 출력됨.
System.out.print(human+10);        // 화면에 출력하되 human에 담긴 100이란 숫자에 10을 더해서 110을 출력함.
int result = human+10;                // result란 변수에 110을 저장함. human(100)+10
```

2.2. 데이터 타입

- 데이터 타입 : 정수타입, 실수타입, 논리타입 이 있음

데이터 타입 특징

값의 종류	기본타입	메모리 사용크기	저장되는 값의 범위
정수	byte	1 byte (8 bit)	$-2^7 \sim (2^7-1)$ 에 해당하는 -128~127 사이의 값 처리 가능
	char	2 byte (16 bit)	$0 \sim (2^{16}-1)$ 에 해당하는 0~65535 사이의 값 처리 가능
	short	2 byte (16 bit)	$-2^{15} \sim (2^{15}-1)$ 에 해당하는 -33768~33767사이의 값 처리 가능
	int	4 byte (32 bit)	$-2^{31} \sim (2^{31}-1)$ 사이의 값 처리 가능
	long	8 byte (64 bit)	$-2^{63} \sim (2^{63}-1)$ 사이의 값 처리 가능
실수	float	4 byte (32 bit)	지수(8 bit) + 가수 (32 Bit) + 부호(1 Bit) 처리 가능 ex) 1.2345는 0.12345×10^1 ➔ 지수 (1), 가수 (0.12345)임
	double	8 byte (64 bit)	지수(11 bit) + 가수 (52 Bit) + 부호(1 Bit) 처리 가능
논리	boolean	1 byte (8 bit)	true, false

체크사항

1. char형은 한글자의 문자를 표현함

```
char human1 = 'A'              // 한문장일 때는 홑따옴표
char human2 = 65
Sytem.out.println (human1); // A가 출력됨
Sytem.out.println (human2);
// human2는 65가 저장되어 있지만 ASC코드인 A가 출력됨
Sytem.out.println ( (int)human2);          // 65 출력
// 정수형으로 형변환시 65 출력됨.
```

2. long형

- 8byte 이상의 큰 수로 초기화 시 L을 붙여야 함.

```
ex) long human3 = 20;                  // 에러 발생하지 않음
    long human4 = 10000000000;  // 에러 발생
    long human5 = 10000000000L;  // 에러 발생하지 않음
```

3. Float형

- 끝에 'F' / 'f' 를 붙여서 사용해야 함.

```
ex) float human6= 3.14 (에러 발생)
    float human7= 3.14F (에러 발생하지 않음)
```

2.3. 데이터 타입의 변환

- 데이터 타입변환 : 데이터 타입을 다른 타입으로 변환하는 것 (byte<➤ int 등)
- 종류 : 자동(묵시적) 타입변환, 강제(명세적) 타입변환

자동 (묵시적) 타입 변환

자동 타입변환

작은 크기의 타입을 큰 크기의 타입에 저장할 때 데이터 타입의 변화가 일어나는 것을 의미함

큰 크기 타입 = 작은크기 타입

저장

ex)

```
byte human1 = 10;
int human2 ;
human2 = human1;
```

byte human1 = 10
(이진수 : 00001010

0	0	0	0	1	0	1	0

Byte형 데이터 타입 (1 byte) 이 Int 형 데이터 타입 (4 byte)으로 자동변환되어 human2에 저장됨

int human2 = 10

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

3 byte 공간에 0으로 채워짐

주의사항

ex)

```
byte human1 = -10;
char human2 ;
human2 = human1;
```

char 타입은 0~65535 의 값을 처리 가능합니다.
음수는 사용불가.
이 때는 에러 발생함.

2.3. 데이터 타입의 변환

- 데이터 타입변환 : 데이터 타입을 다른 타입으로 변환하는 것 (byte<➤ int 등)
- 종류 : 자동(묵시적) 타입변환, 강제(명세적) 타입변환

강제 (명시적) 타입 변환 (casting)

강제 타입변환

큰 크기의 타입을 작은 크기의 타입에 저장할 때 강제적으로 처리하는 프로그램 기법

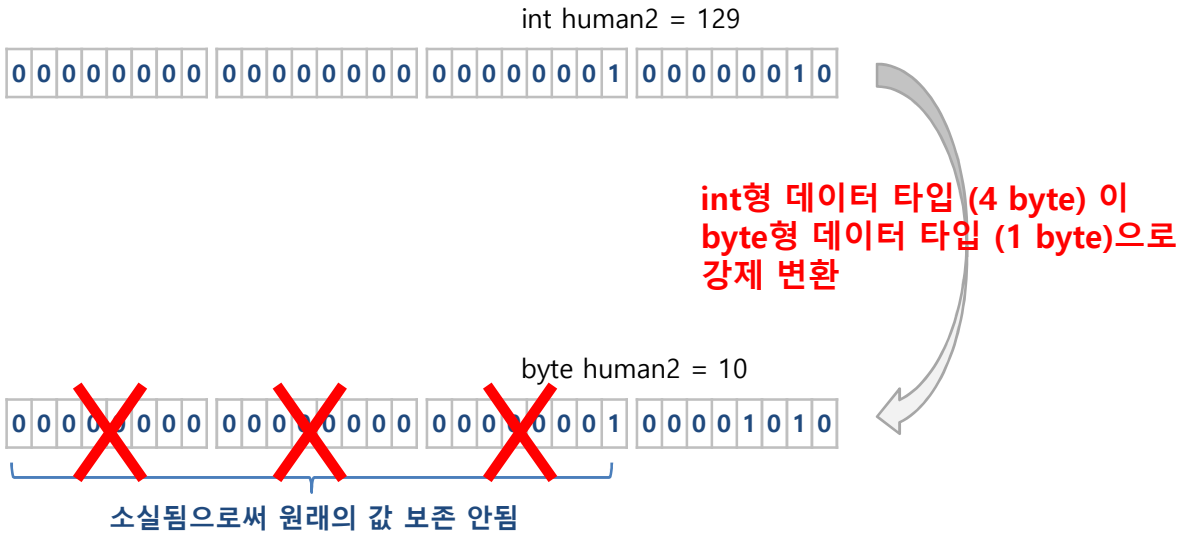
작은 크기 타입 = (작은크기 타입) 큰 크기 타입

쪼갠 후 저장

ex)

```
int human1 = 129;
byte human2 ;
human2 = (byte) human1;
```

human2를 출력하면
129가 아닌 다른 값이 나옴



3장. 연산자

3.1. 연산자

- 연산자 : 산술, 부호, 문자열, 대입, 증감, 비교, 논리, 조건 등의 연산을 하는 기준
- 연산식 : 연산자를 통해 각 변수 또는 상수가 연산을 하는 식

특징

연산자

연산자의 종류	연산자	기능설명	예 시
산술연산자	+, -, *, /, %	사칙연산, 나머지 계산	
부호연산자	+, -	양수, 음수의 부호	Int a = -10 ➔ 음수 10을 a의 변수에 저장함
문자열연산자	+	문자열 연결	
대입연산자	=, +=, -=	우변의 값을 좌변에 대입함	human += 1 ➔ human = human+1
증감연산자	++, --	1 증가 및 감소	human++ ➔ human = human + 1
관계연산자	<,>, =>, =<,	값의 관계를 비교함	
논리연산자	&&, ||, ^, !	and, or 및 논리부정 반환	
조건연산자	(조건)? A : B	조건이 참이면 A, 거짓이면 B 수행	

연산우선순위

1. 왼쪽에서 오른쪽으로 연산 수행 (일반적)

int human = 100 * 100 / 5

연산 순서 →

➔

human = (100*100) / 5 = 10000/5 = 2000

2. 대입연산자의 경우 오늘쪽에서 왼쪽으로 수행함

human1 = human2 = human3 = 100

← 연산 순서

➔

human3=100

human2 = human3

human1= human2

3.2. 증감연산자

- 증감연산자 : 변수의 값을 1증가 또는 감소시키는 연산자
- 주의사항 : 순서를 확인함에 있음

증감연산자의 특징

문법

human++ ++human } human = human + 1	human-- --human } human = human - 1

주의사항

1. ++human의 경우 증감을 우선한 후 다른 연산자와 계산함

```
int human = 99;
int result = ++human * 10;
System.out.println ("human="+human);
System.out.println ("result="+result);
```

➔

human = 100
result = 1000

++human을 통해 human에는 100이 우선 처리됨.
그래서 result = 100 * 10 = 1000

2. human++ 의 경우 다른 연산자와 계산 후 증감함

```
int human = 99;
int result = human++ * 10;
System.out.println ("human="+human);
System.out.println ("result="+result);
```

➔

human = 100
result = 990

human++은 (human* 10)를 연산한 이후에 처리됨.
그래서 result = 99 * 10 = 990 이 되고,
Human은 100이 된다.

3.3. 논리부정연산자 / 문자열연결연산자

- 논리부정연산자 : true ➔ false로, false ➔ true로 변환함 (boolean 타입만 사용가능)
- 문자열 연결연산자 : 문자열끼리 연결시킴 (String 타입에서 사용)

특징

논리부정 연산자

! true ➔ false
! false ➔ true

```
boolean human1 = false;
boolean human2 = !human1;   // !human1은 false에서 true로 변경됨

System.out.println (human2);  //  true가 출력됨
```

! 는 Not을 의미함

문자열연결연산자

❖ 2개 이상의 연산자를 연결함

```
String human1 = "휴먼";
String human2 = "교육";
String human = human1 + human2 + "센터";

System.out.println (human);  //  "휴먼교육센터" 가 출력됨
```

3.4. 관계연산자 / 논리연산자

- 관계연산자 : 2개의 값의 관계를 비교를 하여 true, false 를 판별함
- 논리연산자 : 2개의 boolean 값의 관계를 비교하여 true, false 판별함

특징

비교관계연산자

❖ '>, <, >=, <='
. 두 개의 피연산자 사이의 크기를 비교하여
  논리형 결과 (true, false)를 반환

ex)
int human = 100;
boolean result= (human >=80) // result은 true

항등관계연산자

❖ '==, !='
. == : 2개의 연산자가 동일할 경우 true 반환
. != : 2개의 연산자가 동일할 경우 false 반환

ex)
int human 100;
boolean result= (human ==80) // result은 false

논리연산자

구분	연산식	결과	설명
AND (논리곱)	true && true	true	2개의 연산자 모두가 true일 경우 true로 반환
	true && false	false	
	false && true	false	
	false && false	false	
OR (논리합)	true || true	true	2개의 연산자 중 true가 있을 경우 true로 반환
	true || false	true	
	false || true	true	
	false || false	false	

구분	연산식	결과	설명
XOR - 배타적 논리합	true ^ true	false	2개의 연산자가 같을 경우 true로 반환 다르면 false 반환
	true ^ false	true	
	false ^ true	true	
	false ^ false	false	
NOT - 논리 부정	! true	false	논리값을 바꿈
	! false	true	

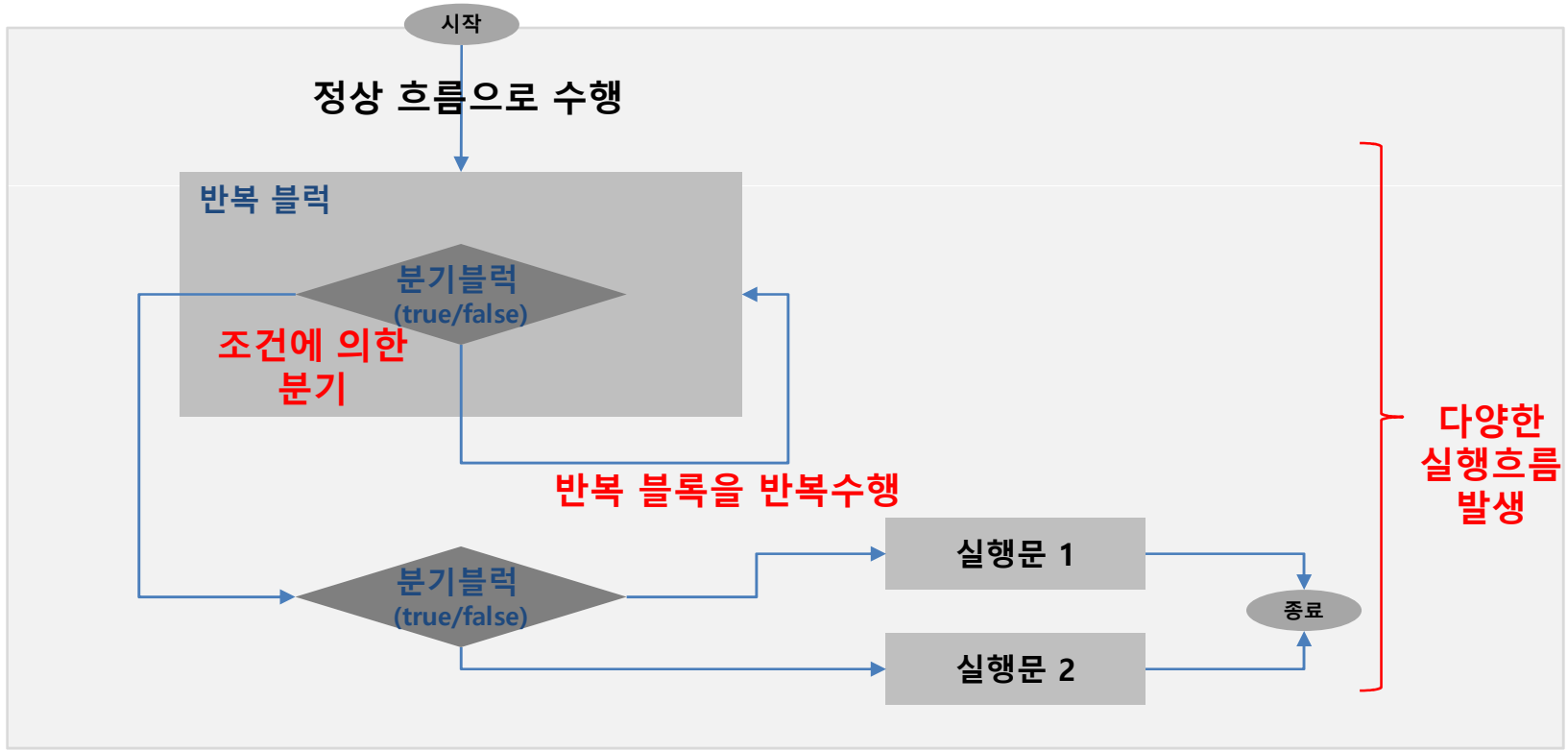
4장. 제어문

4.1. 코드 실행 흐름 이해

- Java 프로그램은 Main 메서드에서 시작하여 순차적으로 실행됨
- 프로그램 제어란 반복문을 통해서 반복을 하고, 분기문을 통해서 분기를 수행하는 것을 의미

코드 실행 흐름

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

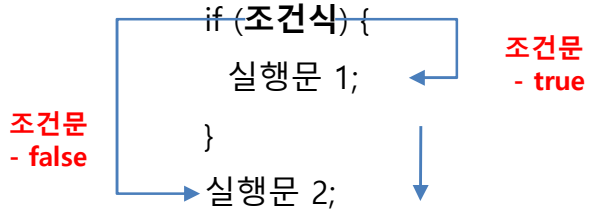


4.2.1. 조건문 (if 문)

- 프로그램의 분기는 조건문의 true,false 의 결과에 따라서 이루어짐.
- 분기문은 중괄호 ({ / })로 이루어진 부분을 시작과 끝으로 인식함.

조건문 (if 문)

프로그램 문법



Case 1 : 조건문이 true일 경우
- 조건문의 중괄호 블록 안에 있는 실행문 1 수행함.
- 그 후 실행문 2가 수행됨

Case 2 : 조건문이 false일 경우
- 실행문 2수행
- 조건의 충족되지 않으므로 실행문 1은 건너뜀

프로그램 사례

예제) 점수가 60점 이상일 경우 합격 판정하는 프로그램

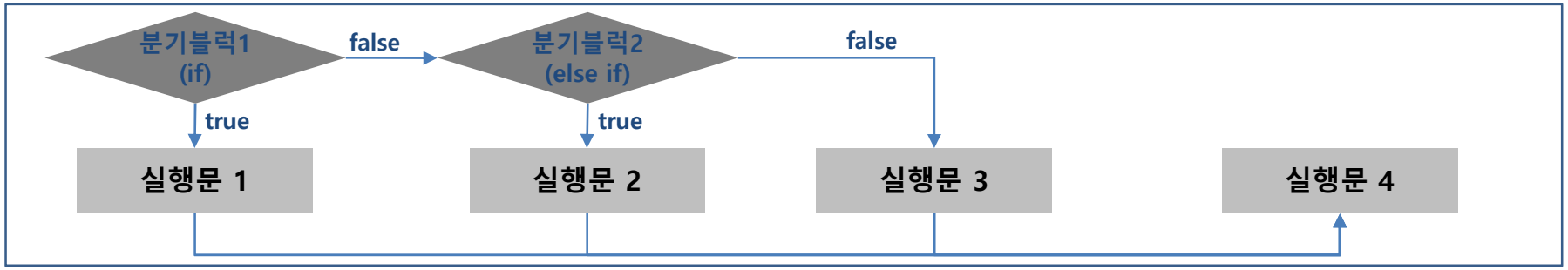
```
public class Human {
  public static void main (String [] args) {
    int kor = 80;
    System.out.println("점수= " + kor);

    if (kor >= 60) {
      System.out.println("점수가 60점보다 높으므로
                          합격입니다.");
    }
    System.out.println("프로그램을 종료합니다.");
```

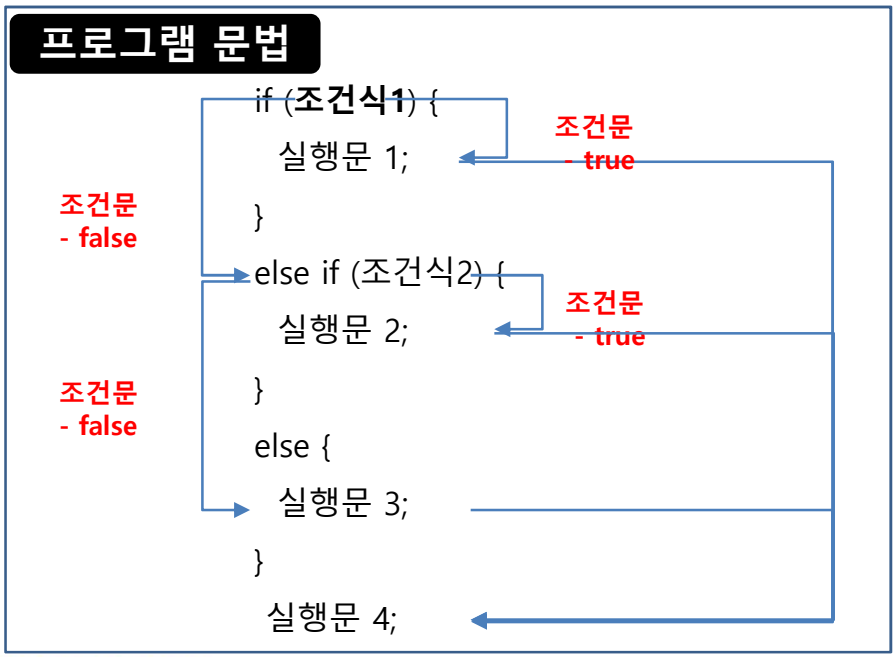
4.2.2. 조건문(if ~ else문)

- 프로그램의 분기는 조건문의 true, false 의 결과에 따라서 이루어짐.
- else는 조건문이 false일 경우 수행되는 제어문임

조건문 (if ~ else문)



프로그램 문법



프로그램 사례

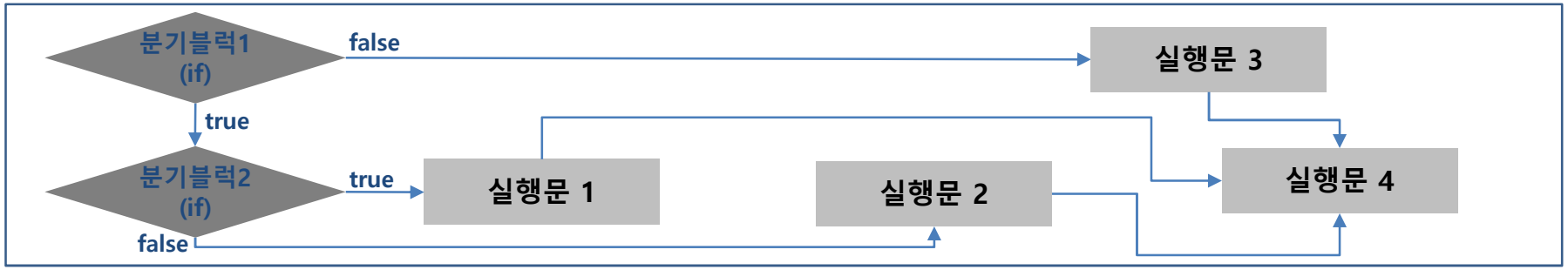
예제) 점수가 90이상이면 A, 80이상이면 B, 그 외는 C학점

```
public class Human {
  public static void main (String [] args) {
    int kor = 80;
    if (kor >= 90) {
      System.out.println("A학점입니다.");
    }
    else if (kor >= 80) {
      System.out.println("B학점입니다.");
    }
    else {
      System.out.println("C학점입니다.");
    }
    System.out.println("프로그램을 종료합니다.");
```

4.2.3. 조건문(중첩 if 문)

- 프로그램의 분기는 조건문의 true,false 의 결과에 따라서 이루어짐.
- if 문 안에 또 다른 if문이 있을 수 있으며, 이 역시 true/false에 따라서 분기가 이루어짐

조건문 (중첩 if 문)



프로그램 문법

조건문 - false

조건문 - true

조건문 - false

조건문 - true

```
if (조건식1) {
  if (조건식2) {
    실행문 1;
  }
  else {
    실행문 2;
  }
}
else {
  실행문 3;
}
실행문 4;
```

프로그램 사례

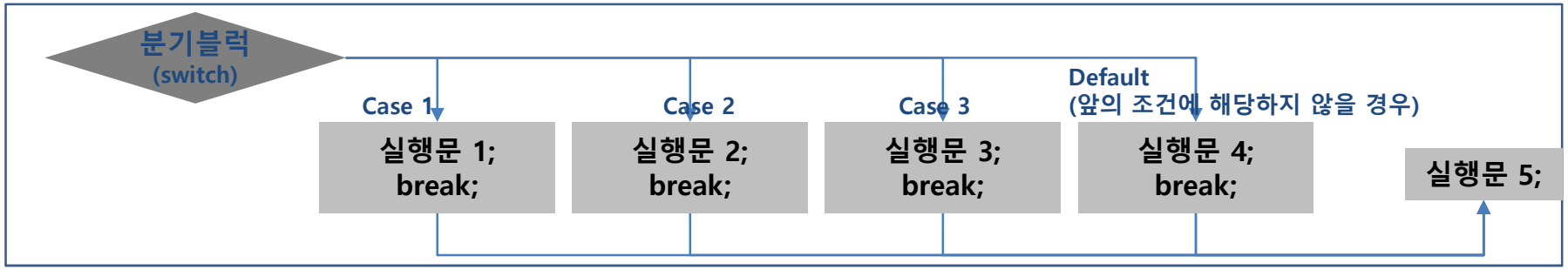
예제) 점수가 60이상이면 합격, 그 중에서도 90점 이상일 경우 장학생여부 판별하는 프로그램

```
public class Human {
  public static void main (String [] args) {
    int kor = 80;
    if (kor >= 60) {
      System.out.println("합격입니다..");
      if (kor >90) {
        System.out.println("당신은 장학생입니다.");
      }
      else {
        System.out.println("장학생이 되지 못한 것이 아쉽습니다.");
      }
    }
    System.out.println("프로그램을 종료합니다.");
}
```

4.2.4. 조건문(swich ~ case 문)

- 분기를 위한 조건이 특정 수치로 정의될 경우 switch ~ case 문을 사용함.
- case 문 처리 이후 break로써 분기문을 마무리 지어야 함.

조건문 (switch ~ case 문)



프로그램 문법

```
switch(변수) {
  case (값1) :
    실행문 1;
    break;
  case (값2) :
    실행문 2;
    break;
  case (값3) :
    실행문 3;
    break;
  default :
    실행문 4;
    break;
}
실행문 5;
```

변수 = 값1

변수 = 값2

변수 = 값3

변수 의 값이 위의 조건에 해당하지 않을때

열거형의 데이터 타입 변수

Break문으로 통해 Switch 조건문 빠져나옴

주의사항 :
각 case별 break문이 생략될 경우 하위로 조건을 추가적으로 탐색함.
그리고 default도 수행됨.

프로그램 사례

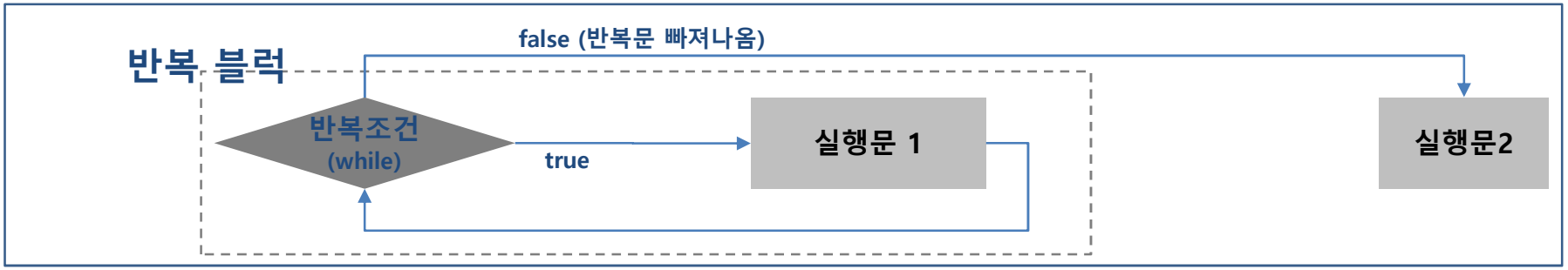
예제) 회원 grade별로 회원을 표시하는 프로그램.

```
public class Human {
  public static void main (String [] args) {
    char  grade = 'A';
    switch (grade) {
      case 'A' :
        System.out.println("우수회원입니다."); break;
      case 'B' :
        System.out.println("일반회원입니다."); break;
      default :
        System.out.println("비회원입니다."); break;
    }
    System.out.println("프로그램을 종료합니다.");
  }
}
```

4.3.1. 반복문(while 문)

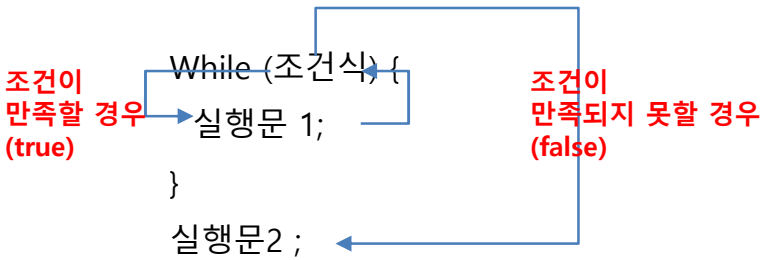
- 반복문은 반복조건이 true일 동안 반복블럭내에서 지속 반복함.
- 반복조건이 false일 경우 반복블럭을 빠져나와 다음 프로그램을 수행함.

조건문 (while 문)



프로그램 문법

반복의 조건을 확인하는 조건문에 의해 반복적으로 실행문을 실행하는 것을 의미한다.



프로그램 사례

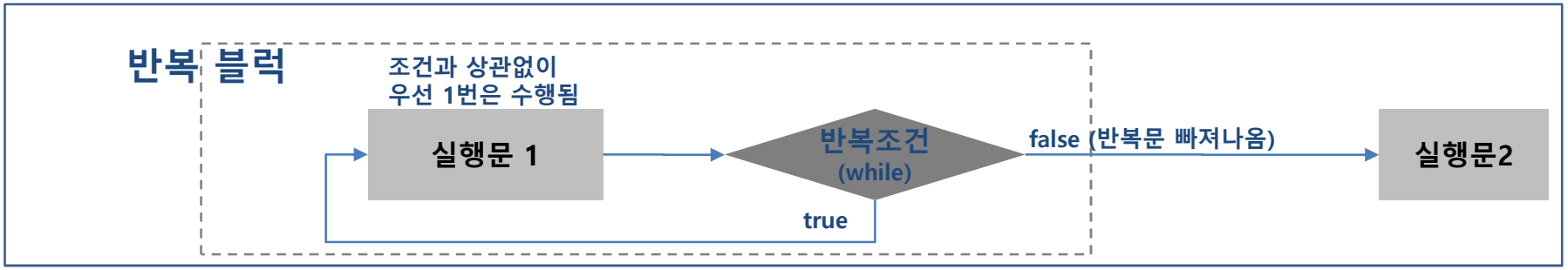
예제) 1~10까지의 누적합산을 처리하는 프로그램

```
public class Human {
  public static void main (String [] args) {
    int total = 0;                  // 누적값을 저장하는 변수
    int index = 1;                  // 반복 인덱스를 저장하는 변수
    while (index <= 10) {
      total = total + index;
      index = index + 1;
    }
    System.out.println("1~10까지의 합산은 = " + total);
  }
}
```

4.3.2. 반복문(do~while 문)

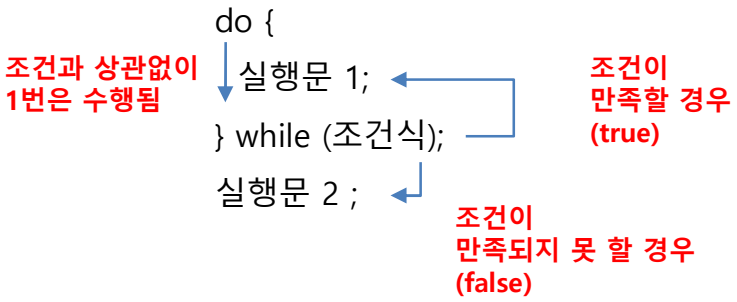
- 반복문은 반복조건이 true일 동안 반복블럭내에서 지속 반복함.
- do 문장에 의해서 우선 한번은 실행한 후 반복은 추후 확인함

조건문 (while 문)



프로그램 문법

while 문과 do~while의 차이점은 do에 의해서 우선 한번은 실행된다는 차이가 있다.
즉, do~while문은 반복의 조건문이 블록 뒤에 있음



프로그램 사례

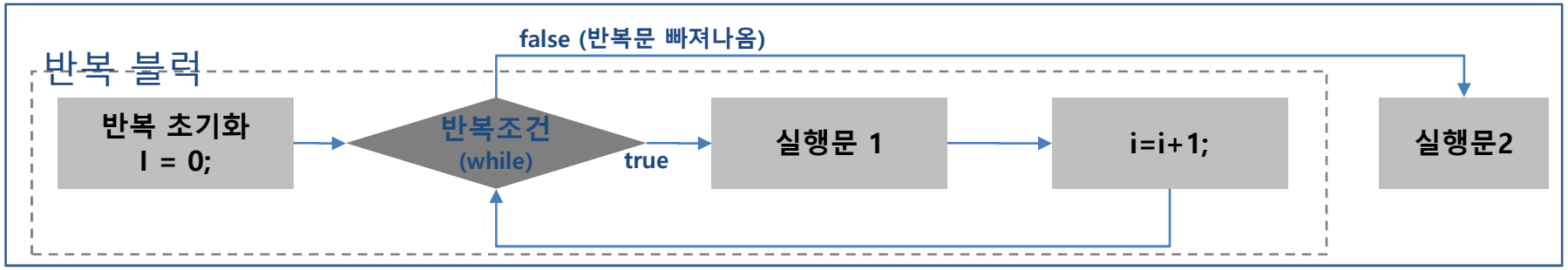
예제) 1~10까지의 누적합산을 처리하는 프로그램

```
public class Human {
  public static void main (String [] args) {
    int total = 0;              // 누적값을 저장하는 변수
    int index = 0;              // 반복 인덱스를 저장하는 변수
    do {
      index = index + 1;
      total = total + index;
    } while (index < 10) ;
    System.out.println("1~10까지의 합산은 = " + total);
  }
}
```

4.3.3. 반복문(for 문)

- 반복문은 반복조건이 true일 동안 반복블럭내에서 지속 반복함.
- 반복조건이 false일 경우 반복블럭을 빠져나와 다음 프로그램을 수행함.

조건문 (for 문)



프로그램 문법

반복문은 반복 index만큼 반복하면서 실행문을 실행하는 일들이 많음. 따라서 이와 같은 경우는 아래의 순서를 따르며, for문을 수행하는 적이 많음

1. 반복 index 초기화
2. 반복 조건문 설정
3. 반복 index의 증감

```
for (index= 0 ; index < 10 ; index=index+1) {
     // 반복 인덱스 초기화
     // 반복의 조건 수행
     // 반복 인덱스의 증감
  실행문 1;
}
실행문 2;
```

프로그램 사례

예제) 1~10까지의 누적합산을 처리하는 프로그램

```
public class Human {
  public static void main (String [] args) {
    int total = 0;                  // 누적값을 저장하는 변수
    for (int index = 1; index <= 10 ; index=index+1) {
      total = total + index;
    }
    System.out.println("1~10까지의 합산은 = ", total);
  }
}
```