

Java Week 1

Min-su Kim

Korea Univ.

October 3, 2018

Overview

① Introduction

1. JDK
2. IDE
3. Why Java?

② Java Basics

1. First Program
2. Variables
3. Operators
4. Flow Control
5. I/O
6. String
7. Array
8. Method

What is JDK?

Java Development Kit(JDK)

- JDK는 자바 애플리케이션과 애플릿을 개발하기 위한 개발 환경 소프트웨어



IDE

What is IDE?

- 통합 개발 환경(Integrated Development Environment)
- 프로그램 개발에 관련된 모든 작업을 한 번에 처리할 수 있게 해주는 소프트웨어

IDE의 종류(for Java)

- Eclipse
- IntelliJ

Why Java?

Programming paradigm

- 객체지향프로그래밍(Object Oriented Programming)

What is OOP?

- 프로그램 설계의 방법 중 하나
- 객체(Object) 개념을 중심으로 프로그램을 설계
- 기존의 프로그램에 대해 확장성이 좋음

OOP Example

Drawing a circle

Circle has ...

- color
- radius
- center
- ...

Extension

- change color
- ...

First Program

Hello, World! 출력하기

- IDE나 코드 편집기를 이용해서 실습
- Class name은 .java 파일 이름과 같아야 합니다.
- 작성한 코드 실행은 Ctrl + F11 (for Eclipse)
- 터미널 or cmd에서의 실행은 다다음 장에..

First Program

```
public class Hello  
{
```

▶ Run |  Debug

```
    public static void main(String[] args)  
    {  
        System.out.println("Hello, World!");  
    }  
}
```


First Program

```
Samsungs-MacBook-Pro:JAVAS stem$ ls
Hello.java
Samsungs-MacBook-Pro:JAVAS stem$ javac Hello.java
Samsungs-MacBook-Pro:JAVAS stem$ ls
Hello.class      Hello.java
Samsungs-MacBook-Pro:JAVAS stem$ java Hello
Hello, World!
Samsungs-MacBook-Pro:JAVAS stem$
```

- ls: 현재 디렉토리 내의 요소들을 출력
- Hello.java: 작성한 자바 소스 코드 파일
- javac Hello.java: 자바 소스코드를 컴파일
- Hello.class: 작성한 자바 소스 코드 파일의 클래스
- java Hello: 작성한 코드를 실행(main 함수가 있는 class의 이름)

Data types

자료형	데이터	크기
boolean	참, 거짓	1 바이트
char	문자	2 바이트
byte	정수	1 바이트
short		2 바이트
int		4 바이트
long		8 바이트
float	실수	4 바이트
double		8 바이트

- 선언 방식은 C와 같음

Data types

```
boolean bool = true;  
char      bloodType = 'A';  
byte     date = 4;  
short    month = 7;  
int      year = 1997;  
long     studentNumber = 2017320168;  
float    PI = 3.141592;  
double   pi = 3.1415926535;
```

- camelCase
- case sensitive

Identifier

identifier: 식별자 = 변수의 이름(name of variable)

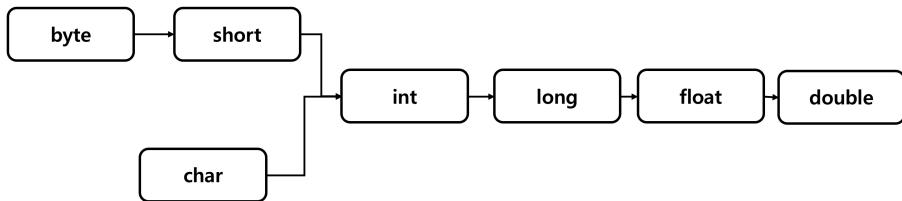
변수명 선언 규칙

- ① 적어도 하나의 문자가 있어야 함
- ② 첫 글자는 알파벳, _, \$로 시작해야 함(숫자 사용 불가)
- ③ 첫 글자가 아닌 경우, 알파벳, 숫자, _, \$ 사용 가능

Example

- `int num;` // 가능
- `int 1num;` // compile error

Type casting



- 화살표 방향으로는 implicit casting 지원
- 반대 방향으로 가려면 explicit casting 필요

Type casting example

```
long    num1 = 1;
long    num2 = 2;
int     sum;

sum = num1 + num2;    /// 컴파일 에러
sum = (int)(num1 + num2);    /// explicit casting
num1 = sum;    /// implicit casting
```

대입, 산술 연산자

연산자	기능
=	오른쪽에 있는 값을 왼쪽의 변수에 대입 <code>int num = 10;</code>
+	두 값을 더함 <code>num = 5 + 4;</code>
-	왼쪽 값에서 오른쪽 값을 뺌 <code>num = 8 - 3;</code>
*	두 값을 곱함 <code>num = 4 * 3;</code>
/	왼쪽의 값을 오른쪽의 값으로 나눔 <code>num = 43 / 7;</code>
%	왼쪽의 값을 오른쪽의 값으로 나눈 나머지 <code>num = 43 % 7;</code>

- /는 두 번째 인자에 0이 올 수 없음(**Run-time error!**)
- % 연산은 실수 타입은 인자로 올 수 없음

Type

연산 결과의 타입

- 정수와 정수의 연산 = 정수
- 실수와 실수의 연산 = 실수
- 실수와 정수의 연산 = 실수

Example

- $10 / 3 = 3$
- $10.0 / 3.0 = 3.3333...$
- $10.0 / 3 = 3.3333...$
- $10.0 + 3 = 13.0$

복합 대입 연산자

복합 대입 연산자	대입, 산술 연산자
<code>a += b;</code>	<code>a = a + b;</code>
<code>a -= b;</code>	<code>a = a - b;</code>
<code>a *= b;</code>	<code>a = a * b;</code>
<code>a /= b;</code>	<code>a = a / b;</code>
<code>a %= b;</code>	<code>a = a % b;</code>

관계 연산자

연산자	기능
<	왼쪽 값이 더 작은가 $n1 < n2$
>	왼쪽 값이 더 큰가 $n1 > n2$
<=	왼쪽 값이 작거나 같은가 $n1 \leq n2$
>=	왼쪽 값이 크거나 같은가 $n1 \geq n2$
==	두 값이 같은가 $n1 == n2$
!=	두 값이 다른가 $n1 != n2$

논리 연산자

연산자	기능
&&	두 값이 모두 참이면 참 true && true => true, true && false => false, false && false => false
	두 값 중 하나만 참이면 참 true true => true, true false => true, false false => false
!	참이면 거짓, 거짓이면 참 !true => false, !false => true

증가, 감소 연산자

연산자	기능
++ (prefix)	변수의 값을 1 증가 후 연산 ++val
-- (prefix)	변수의 값을 1 감소 후 연산 --val
++ (postfix)	연산 후 변수의 값을 1 증가 val++
-- (postfix)	연산 후 변수의 값을 1 감소 val--

삼항 연산자

(boolean type expr) ? (expr1) : (expr2);

- 첫 인자에는 true / false를 반환하는 boolean type의 expression 필요
- 만약 첫 번째 인자가 참이면 expr1 반환
- 만약 첫 번째 인자가 거짓이면 expr2 반환

Example

- (5 > 3) ? 5 : 3; // 5
- (3 > 5) ? 5 : 3; // 3

Operator Priority

우선순위	연산자	결합방향
1	[], ,	=>
2	expr++, expr--	<=
3	++expr, --expr, +expr, -expr, !, (type)	=>
4	*, /, %	=>
5	+, -	=>
6	<, >, <=, >=	=>
7	==, !=	=>
8	&&	=>
9		<=
10	? expr : expr	<=
11	=, +=, -=, *=, /=, %=	<=

Quiz1

```
int    val1 = 5;  
int    val2 = 5;
```

```
System.out.println(++val1);  
System.out.println(val2++);
```

- 1 출력되는 값
- 2 val1과 val2에 들어있는 값

Quiz1 answer

① 6
5

- ++val1는 값을 증가시킨 후에 출력하므로 6 출력
- val2++는 출력한 후에 값을 증가시키므로 5 출력

② 모두 6이 들어있음

Quiz2

```
System.out.println('a' + 1);
```

- 출력 값

Quiz2 answer

- 98 출력
- 'a'의 아스키 코드 값은 10진수로 97
- (char) + (int)인데, implicit casting으로 결과 값은 (int)
- 'b'를 출력하고 싶으면 ...
- `System.out.println((char)('a' + 1));`

주석

- 한줄 주석 `// 주석`
- 여러 줄 주석 `/* 주석 내용 */`

If statement

```
if(boolean type expr)
{
    statements
    ...
}
```

C와 다른 점?

- C에서는 0이 false, otherwise true
- boolean type이 따로 있으므로 조건문 내에는 반드시 boolean type expression 필요
- 조건 내에 boolean 외의 type이 들어가면 오류

switch case

```
switch (case를 나눌 변수)
{
    case n1: // 변수의 값이 n1일 경우
        ...
        break;
    default: // 해당하는 변수 값이 없을 경우
        ...
}
```

- C와 같음

while loop

```
while (boolean expr)
{
    statements;
}
```

실행 순서

- ① 조건문 검사
 - ② statements 실행
 - ③ 1번부터 다시
- C언어와 같음

do while loop

```
do  
{  
    statements;  
} while(boolean expr);
```

실행 순서

- ① statements 실행
- ② 조건 검사
- ③ 1번부터 다시

for loop

```
for (초기화; 조건식; 증감식;) {  
    statements;  
}
```

실행 순서

- 1 초기화
 - 2 조건식
 - 3 statements
 - 4 증감식
 - 5 2번부터 다시
- C언어와 같음

for each loop

```
for (변수명 : 배열 or 문자열 or ...)  
{  
    statements;  
}
```

- : 뒤의 것을 하나씩 변수명에 담아서 씀
- 나중에 배열 공부한 후에 사용해보겠습니다.

Quiz3

```
int    x = 5;

if(x > 10)
    if(x > 100)
        System.out.println("big enough!");
else
    System.out.println("not that..");
```

Quiz3 answer

-
- 아무 것도 출력되지 않음
- 당연한 결과

Quiz3 answer

```
int    x = 5;

if (x > 5)
    System.out.println("bigger than 5");
else if (x < 5)
    System.out.println("smaller than 5");
else
    System.out.println("5");
```

- else if문?
- 결국 else 뒤에 한 문장인 if가 붙어서 형성됨
- 만약 else가 위의 if에 붙는다고 하면 else if 문은 쓸 수가 없음
- so, 당연한 결과

continue, break

- continue
 - 그 반복의 맨 끝으로 감
- break
 - 가장 가까운 반복문을 끝냄

infinite loop

각 loop 문의 조건에 항상 true인 값을 넣어주면 됨

Input

C언어에서는 scanf() 함수를 이용해 키보드의 입력을 받음.

then, how about java?

Scanner 객체를 사용

다음 장 예제 코드

Scanner

```
import java.util.Scanner;

public class Scan
{
    ▶ Run | 🐛 Debug
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);
        int num;

        num = keyboard.nextInt();
        System.out.println(num);

        keyboard.close();
    }
}
```


Scanner

- Scanner 객체를 사용하려면 import로 불러와야 함
- `import java.util.Scanner;` 또는 `import java.util.*;`
- 정수를 받으려면 `(Scanner객체 변수명).nextInt();`

Scanner

Scanner 클래스의 메소드

- `public boolean nextBoolean()`
- `public int nextInt()`
- `public long nextLong()`
- `public float nextFloat()`
- `public double nextDouble()`
- `public String nextLine()`
- etc...

String

C에서는 문자열을 char의 배열로 사용
but, java는 String이라는 class가 존재
선언 방법

- `String 변수명 = new String("Hello");`

축약 가능

- `String 변수명 = "Hello";`

두 개는 살짝 다름

String compare

String 비교에는 `==`이 아닌 자체 함수인 `.equals()`를 사용

String compare

```
if ("abc" == "abc")  
    System.out.println("same");  
else  
    System.out.println("diff");
```

- same

String compare

```
String str1 = new String("abc");  
String str2 = new String("abc");  
  
if (str1 == str2)  
    System.out.println("same");  
else  
    System.out.println("diff");
```

- diff

String compare

```
String str1 = new String("abc");  
String str2 = new String("abc");  
  
if (str1.equals(str2))  
    System.out.println("same");  
else  
    System.out.println("diff");
```

- same

String concatenate

C언어에서는 string.h헤더 include 후, strcat()함수 사용 but...
java에서는 +로 해결

- "abc" + "cde" == "abccde"

또는 concat()메소드 사용

- "abc".concat("cde");

어떤 타입을 문자열과 더하면 결과는 문자열

- "Hello" + 1 = "Hello1"
- "StudentNumber: " + 2017320168 = StudentNumber: 2017320168

Quiz4

```
System.out.println("abc" + 1 + 2);
```

- 출력 값?

Quiz4 answer

- abc12
- abc30이 나오려면 "abc" + (1 + 2)로 해야함

Array

- 같은 타입의 여러 변수를 한 번에 관리할 수 있는 오브젝트
- 지금까지 배운 자료형 모두로 사용 가능
- 심지어 객체 타입도 배열로 선언 가능

Array Example

```
int[]    num1 = new int[6];  
int[]    num2 = new int[]{1, 2, 3, 4, 5, 6};  
int      num3[] = {1, 2, 3, 4, 5, 6};  
  
String   str[] = new String[6];  
for (int i = 0; i < str.length; ++i)  
    str[i] = new String("Hello" + i);
```

- (자료형)[] (배열명) = new (자료형)[(배열의 길이)]
- 대괄호는 자료형 뒤나 배열명 뒤에 붙임
- 객체 배열 선언시 주의!

Array length

(배열명).length;

- 배열의 길이를 반환
- String의 length인 (문자열명).length()와는 다름

for each statement

```
for ((자료형) 변수명: 배열명)
{
    statements;
}
```

- 배열이나 문자열 내의 모든 요소에 순차적으로 접근하기 위해 사용
- 변수명은 for문 내에서 배열의 각 요소에 접근하기 위해서 임시로 사용
- for each문에서는 배열 내의 값을 참조하는 것
- 변수의 값을 바꾼다고 해도 배열 자체의 값이 바뀌는 것이 아님

for each statement example

```
int[]    num = {1, 2, 3, 4, 5, 6};  
  
for(int i = 0; i < num.length; ++i)  
    System.out.println(num[i]);
```

for each statement example

```
int[]    num = {1, 2, 3, 4, 5, 6};  
  
for (int tmp: num)  
{  
    System.out.println(tmp);  
}
```


for each statement example

- 1
- 2
- 3
- 4
- 5
- 6

for each statement 

```
int    num[] = {1, 2, 3, 4, 5};
```

```
for (int i = 0; i < num.length; ++i)  
    num[i] += 1;;
```

for each statement 비교

```
int    num[] = {1, 2, 3, 4, 5};  
  
for (int tmp : num)  
    tmp += 1;
```

- 앞 장의 코드와 다른 코드.
- 앞 장의 코드는 num의 값에 영향을 미침
- But, 이 코드는 num의 값에 직접적인 영향 없음

for each statement example

```
int    num[] = {1, 2, 3, 4, 5};

for (int i = 0; i < num.length; ++i)
{
    int    t = num[i];
    t += 1;
}
```


- 첫 장의 코드와 같은 코드.

Method

- 특정 기능을 수행하는 단위
- C언어의 함수와 비슷한 개념
- class 내부에 선언

Method

```
public class Hello  
{
```

▶ Run |  Debug

```
    public static void main(String[] args)  
    {  
        System.out.println("Hello, World!");  
    }  
}
```

- main 메소드
- println 메소드

Return type

- C언어에서처럼 반환하는 값의 타입.
- void, int ,double etc..
- return = 값의 반환, 메소드의 종료

Recursion

- recursion(재귀): 메소드가 스스로를 호출
- 종료 조건이 반드시 필요

Recursion example

```
public static void main(String[] args)
{
    System.out.println(factorial(5));
    System.out.println(factorial(7));
}

static int factorial(int n)
{
    if (n <= 1)        // ! 종료 조건
        return 1;
    else
        return n * factorial (n - 1); // ! Recursive call
}
```

- 120
- 5040

Scope

- 변수에 접근할 수 있는 영역
- 한 영역 안에서 같은 이름의 변수는 선언 불가능
- 영역? = {}
- for문과 메소드의 경우
- ()안에서 선언된 변수는 따라오는 {} 내에서만 접근 가능

Scope example1

```
int    num[] = {1, 2, 3, 4, 5};  
  
for (int i = 0; i < num.length; ++i)  
    System.out.println(num[i]);  
System.out.println(i);
```

Scope example2

```
public static void increase(int num)
{
    num++;
    System.out.println(num);
}

public static void main(String[] args)
{
    int    num = 5;
    increase(num);
    System.out.println(num);
}
```