자바 기초

자바의 정석

1. 자바
   (1) 자바란?
2. 변수
   (1) 변수란?
   (2) 변수의 타입
   (3) 형 변환
3. 연산자
   (1) 연산자란?
   (2) 연산자 종류(단항, 산술, 비교, 논리, 그 외)
4. 재어문(조건문과 반복문)
   (1) if, switch
   (2) for, while, do-while
5. 배열
   (1) 배열이란?
   (2) 배열 사용 방법
6. 메서드(함수)
   (1) 메서드란?

핵심만 설명!!

(들어가기 앞서) 특강 진행 방법

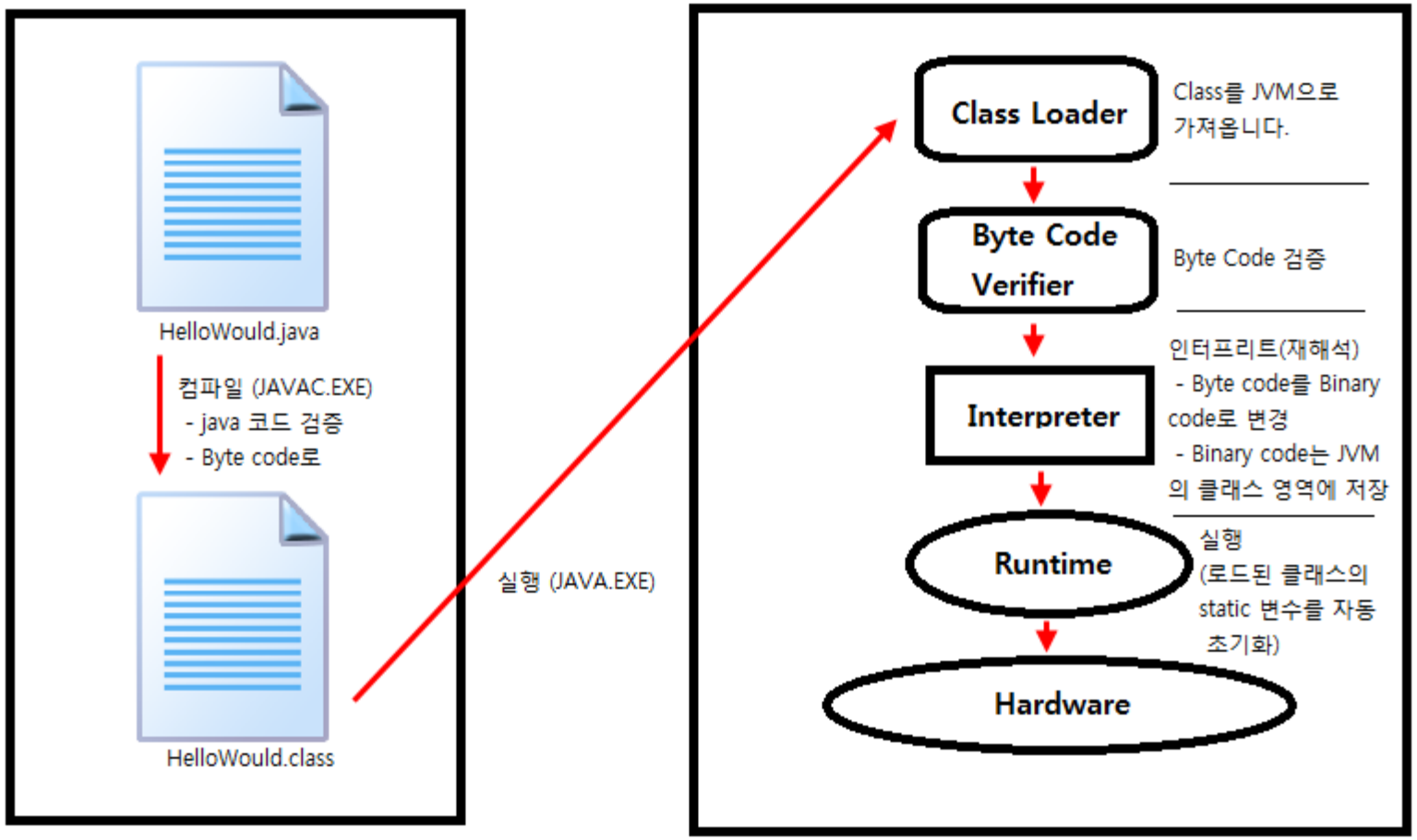
- #1 단원 소개 : 이론 설명
- #2 참고 자료 : 이론 보충 설명(PDF)
- #3 기존 예제 풀이 : JavaExample Project
- #3 기존 과제 풀이 : JavaReport Project
- #4 연습문제 코드 풀이 : 단원 연습문제
- #5 Q & A : 관련 질문과 답변

1. 자바

(1) 자바란?

[특징]

1. 운영체제에 독립적
2. 객체지향언어
3. 자동 메모리 관리
4. 멀티 스레드 지원
5. 동적 로딩
   : 필요한 시점에 클래스 로딩

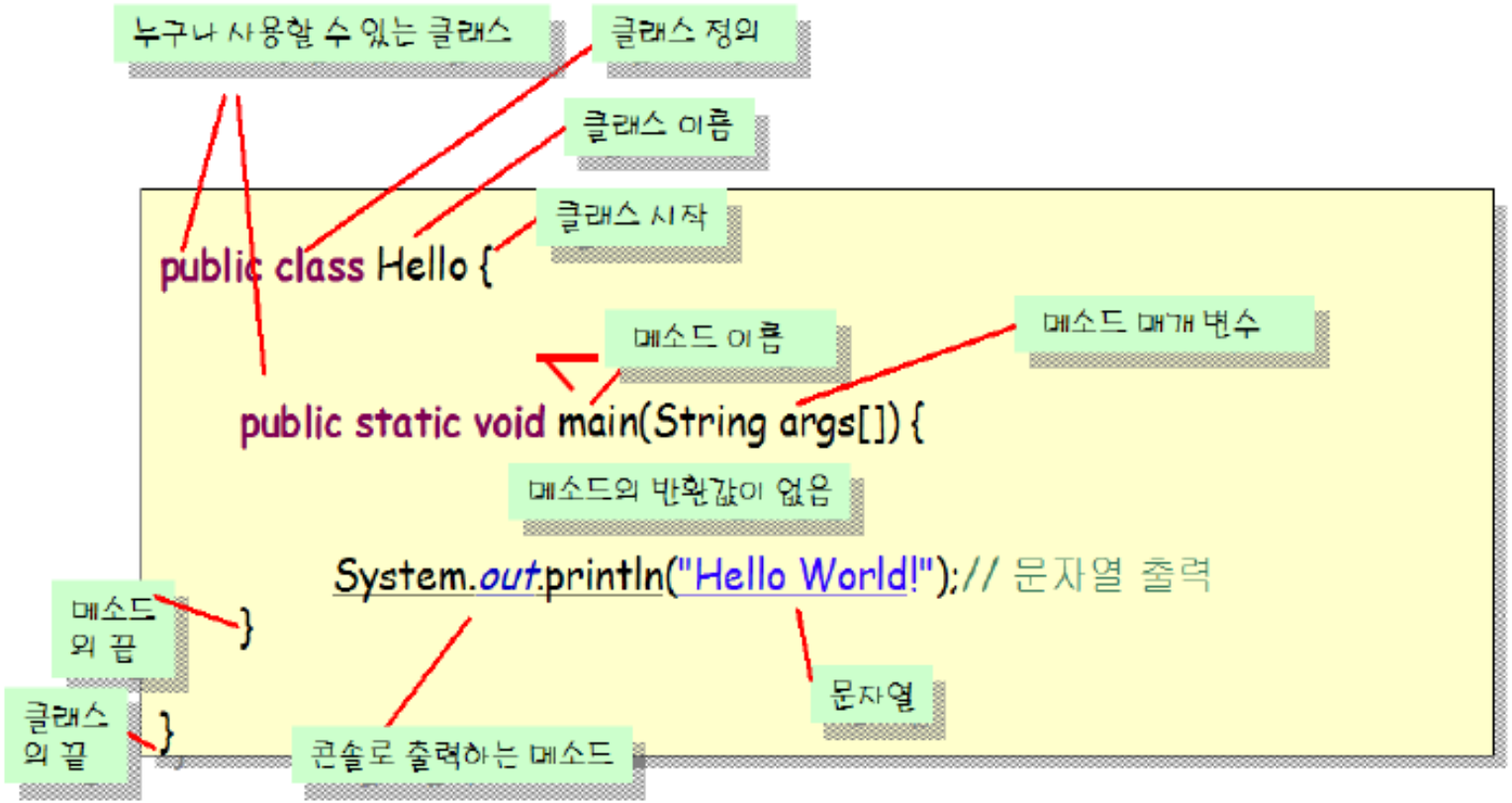


1. 자바

(1) 자바란?

[특징]

1. 운영체제에 독립적
2. 객체지향언어
3. 자동 메모리 관리
4. 멀티 스레드 지원
5. 동적 로딩
   : 필요한 시점에 클래스 로딩



+ 이클립스 단축키

- 컴파일 & 실행 : **Ctrl + F11**
- 주석 : **(해당 영역 선택(여러 줄 가능) 후) Ctrl + /**
- System.out.println() : **syso(타이핑 후) Ctrl + SpaceBar**
- 라인 복사 : **(해당 영역 선택(여러 줄 가능)후) Ctrl + Alt + 방향키(위/아래)**
- 라인 이동 : **(해당 영역 선택(여러 줄 가능)후) Alt + 방향키(위/아래)**
- 라인 삭제 : **(해당 영역 선택(여러 줄 가능)후) Ctrl + D**
- 에디터 화면(크게/작게) : **Ctrl + M**
- 프로젝트 or 클래스 생성 : **Ctrl + N**
- 현재 소스파일 저장 : **Ctrl + S**
- 열린 전체 소스파일 저장 : **Ctrl + Shift + S**

2. 변수

(1) 변수란? - 1

- 변수 : 값을 저장할 수 있는 공간(단 하나의 값을 저장 할 수 있는 공간
- 변수 선언

변수 ?
상수 ?
리터럴 ?

int num = 100;				
int	num	=	100	;
데이터 형 (숫자형)	변수명 (자유롭게 만듦)	대입 연산자	변수에 넣을 값	문장의 완성을 알리는 기호
l - value(left value)		=	r - value(right value)	

이 부분은 변수만 사용가능		오른쪽 값을
		왼쪽 값에
		저장한다.
			이 부분은
			변수 : 값을 저장할 수 있는 메모리상의 공간
			리터럴 : 직접 나타내는 데이터 값(고정값) ex) 100, 200, 300, 1, 3 ...
			상수 : 변하지 않는 값(final으로 정의한 값) ex) final int MAX = 100; // 이때 변수명은 대문자
			연산식 : '+,-.*,/' 등의 연산자를 사용한 식 ex) 100 + 200, 10*20
			사용가능	

2. 변수

(1) 변수란? - 2

- 키워드, 예약어 : 변수나 메서드에서 이름을 지정할 때, 사용하면 안 되는 이름!

객체 관련	class, interface, enum
상속 관련	extends, implements
타입 관련	byte, short, int, long, float, boolean, char, double
제어문 관련	if, switch, else, default, while, do, break, continue
예외처리 관련	try, catch, finally
제어자(접근) 관련	public, protected, private
제어자(접근) 활용 관련	static, final, abstract
연산자 관련	instanceof
생성자 관련	new

[변수 이름 정하는 규칙]

1. 대소문자가 구분되며 길이에 제한이 없다.
2. 예약어를 사용해서는 안 된다.
3. 숫자로 시작해서는 안 된다.
4. 특수문자는 '_'와 '$ ' 만을 허용한다

2. 변수

(2) 변수의 타입

<table>
<tr><td>기본형</td><td>데이터를 저장하고 처리하는 타입
<table>
<tr><td rowspan="2">숫자형</td><td>정수형</td><td>byte(1), short(2), int(4), long(8)</td><td>0, 1, 10, 10L(long형) ...</td></tr>
<tr><td>실수형</td><td>float(4), double(8)</td><td>0.0, 1.0F(float형), 100.0 ...</td></tr>
<tr><td>문자형</td><td>문자열</td><td>char(2)</td><td>'a', 'b', 'c', 'd' ..</td></tr>
<tr><td>대수형</td><td>참 & 거짓</td><td>boolean(1)</td><td>true, false</td></tr>
</table></td></tr>
<tr><td>참조형</td><td>처리하게 될 데이터의 주소 값을 가지고 처리하는 타입
종류 : 배열, 클래스, 인터페이스</td></tr>
</table>

- 초기화

처음 변수를 선언하면 쓰레기 값(정의 되지 않은 값)이 저장되어있다. 만약 이 변수를 += 같은 구문으로 사용하려 할 때, 쓰레기 값에 더하려는 값을 더하게 된다. (자바는 커파일오류 발생)따라서 0의 값을 변수에 처음 저장을 함으로써 나중에 사용할 때, 오류를 줄여준다. 초기화는 웬만하면 처음에 변수를 선언 할 때, 꼭!! 해주는 게 좋다.

2. 변수

(3) 형 변환(Type Casting)

자동형 변환	데이터 형의 크기가 작은 쪽에서 큰 쪽으로 가면 데이터 손실이 없이 변환이 된다.
	반대로 큰 쪽에서 작은 쪽으로 가면 데이터 손실이 생기면서 변환이 된다.
	ex) 데이터 손실 발생 O : byte a = int b; // int형 크기가 더 크기 때문
	데이터 손실 발생 X : int b = byte a; // byte형 크기가 작기 때문

강제형 변환	자동형 변환을 하지 않고, 직접 변환을 원하는 것으로 하고 싶을 때 하는 변환 이것 또한 자동형 변환과 같이 데이터 손실이 발생 할 수도 있고, 발생하지 않을 수도 있다. ex) int rand = (int)(Math.random()); // 원래 Math.random() 함수는 double형

long : 0(L)
float : 0(f)
double : 0(.0)
String : “”
char : ‘’ (오류!!)

3. 연산자

(1) 연산자란?

어떠한 기능을 수행하는 기호(+,-,*,/ 등)

a != b;	a와 b는 같지 않다.
a == b;	a와 b는 같다
+, -, *, /	a=a+b, a=a-b, a=a*b, a=a/b
a%b; (나머지 연산자)	a를 b로 나눈 값의 나머지
>, <, >=, <=	크기 비교
a&&b (AND 연산자)	a도 참이고, b도 참이다
a||b (OR 연산자) **| 기호는 shift키 + ₩키(backspace 왼쪽 키)**	a가 참이거나, b가 참이다

항1 ? 항2 : 항3 (삼항 연산자)	항1 (조건문)
	- 참 : 항2 결과값 이용
	- 거짓 : 항3 결과값 이용
	ex)
	int a=3;
	int result=0;
	result = (a<3) ? 1 : 2;

a++;	a = a+1;

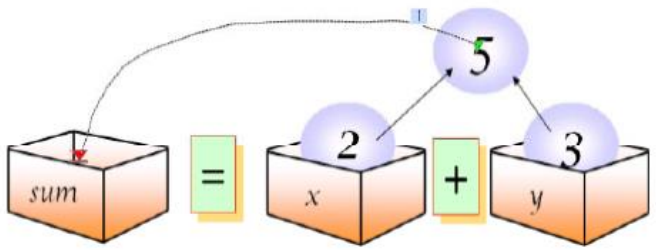
b = ++a;
(Java의 정석 SubNote : p29 참고)
	단항일 경우는 a++과 결과값이 같다
	이항일 경우는 a++과 결과값이 다르다
	int a = 1;
	int b = **a++**;
	System.out.println(b); // 1
	a = 1;
	b = **++a**;
	System.out.println(b); // 2
	이항일 경우는 **++의 위치에 따라** 의미가 달라진다.
	앞에 있을 경우는 **먼저 더한다**는 의미
	뒤에 있을 경우는 **수행 후 더한다**는 의미

a--;	a = a-1;
b = --a;	b = ++a; 와 의미는 같으나 단지, 부호가 다르다.
a+=b;	a = a+b;
a-=b;	a = a-b;
a*=b;	a = a*b;
a/=b;	a = a/b;

3. 연산자

(2) 연산자 종류

(예) sum = x + y;



우선순위 높음

단항 연산자	- 부호 : +, - - 증감 : ++, -- - 부정 : ! - 비트연산자(XOR) : ~
이항 연산자	- 산술 : +, -, *, /, %, <<, >>, >>> (초록색은 생략) - 비교 : >, <, <=, >=, ==, != - 논리 : &&, ||, &, ^, | (초록색은 생략)
삼항 연산자	(조건식) **?** (참) **:** (거짓)
대입 연산자	l-value **=** r-value

우선순위 낮음

4. 제어문(조건문)

(1) if, switch

switch → if (모두 가능)

if → switch (일부 가능)

```
if(조건식){
문장;
}
else if(조건식){
문장;
}
else{
문장
}
```

```
if(a==1){
  print("1번");
}
else if(a==2){
 print("2번");
}
else{
 print("?");
}
```

```
switch(조건식){
case 값1 :
문장
break;
case 값2 :
문장
break;
default :
문장
(break;)
}
```

```
switch(num){
case 1 :
print("1번");
break;
case 값2 :
print("2번");
break;
default :
print("?");
(break;)
}
```

4. 제어문(반복문)

(2) for, while, do-while

for : 반복하게 될 횟수를 미리 알 수 있을 때, 몇 회 반복해라

while : 반복횟수는 중요하지 않고, 어떠한 조건이 반복되는 동안 계속 반복 되도록 할 때 사용

break; : 반복문을 탈출

continue; : 반복문의 끝으로 이동

for 문

```
for(초기식 ; 조건식 ; 증감식){
문장;
}
```

```
for(int i=0 ; i < 10 ; i++){
print(i);
}
```

for each 문

```
for(변수 : 배열){
 print(변수);
}
```

```
for(int data : array){
 print(data);
}
```

```
while(조건식){
문장;
}
```

```
int i=0;
while( i < 10 ){
print(i);
i++
}
```

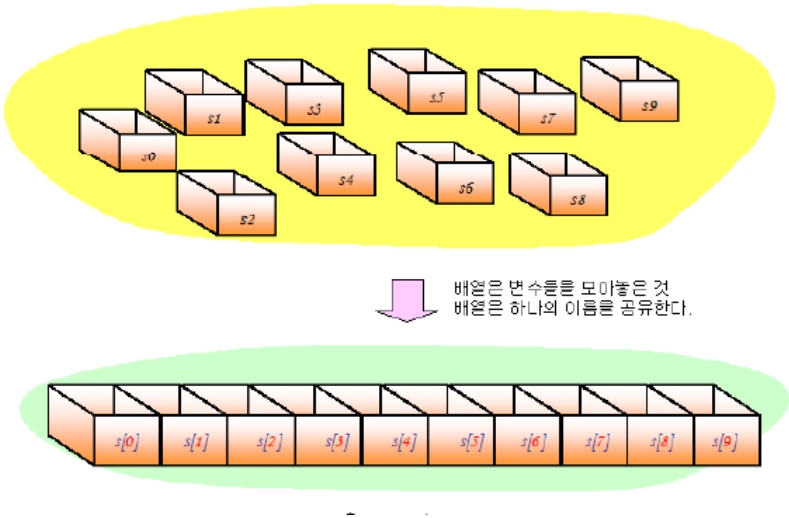
```
do{
 최소 1번 수행
} while(조건식);
```

5. 배열

(1) 배열이란?

- **동일한 타입의 데이터를 하나의 집합으로 다룰 수 있게 사용 되는 자료구조**
  - 어떠한 타입의 데이터를 다룰 것인가?
  - 최대 몇 개의 데이터들을 다룰 것인가?

배열(array): 같은 타입의 변수들의 모임



- **배열 변수 선언** : 데이터타입[] 변수명 → int[] arry
- **배열 생성** : new 데이터타입[크기] → new int[5]
- **배열 변수 = 배열 생성** : int[] arry = new int[5];
- **배열 크기** : 배열 변수.length → arry.length
- **배열의 인덱스(순서)** : 0번 부터 시작

5. 배열

(2) 배열 사용 방법

- **초기화**

  - int[] score = { 1, 2, 3, 4, 5 };
  - int[] score = new int[] { 1, 2, 3, 4, 5 };
  - int[] score;
    score = { 1, 2, 3, 4, 5 }; **// error!!, 배열 변수와 {} 만 사용 할 경우, 배열 변수를 정의 할 때만 사용가능**
  - int[] score;
    score = new int[] { 1, 2, 3, 4, 5 };

- **배열에 값 저장**

  - 인덱스(순서) : score[3] = 100; // 결과 : int[] score = { 1, 2, 3, 100, 5 }; 와 같음

  - for구문 사용시 : for(int i=0; i<5 ; i++) score[i] = 1; // 결과 : int[] score = { 1, 1, 1, 1, 1 }; 와 같음

6. 메서드(함수)

(1) 메서드란?

사용방법은
예제로 확인!!

- 메서드는 **어떤 기능**을 제공하는 프로그램의 단위이다.
- 메서드 생성 방법

#1 만들고자 하는 메서드의 기능을 설명 할 수 있는 메서드 명을 정한다.

(ex) getNumber(), setNumber()

#2 메서드로 보낼 데이터 값(매개변수=파라미터=인수)을 선언 (없으면 공백으로 설정)

(ex) getNumber(), setNumber(int num1, int num2)

#3 메서드를 모두 수행 한 후 데이터 값을 다시 받아 올 것인지 여부를 결정

(ex) 데이터 값을 다시 받아 올 경우 : int getNumber() { return 데이터 값; }

(ex) 데이터 값을 받아오지 않을 경우 : void setNumber(int num1, int num2){ return; }

#4 메서드의 접근권한(제어자)을 결정

(ex) public int getNumber() { return 데이터 값; }

(ex) private void setNumber() { return; }

Q & A !!