1. 자바 시작하기

* 컴퓨터는 2진수로 된 기계어로 동작
* 기계어는 사람이 사용하기 불편하고 복잡하여 프로그래밍 언어가 개발됨
* 프로그래밍 언어로 작성된 소스코드가 컴파일러를 통해서 기계어로 변환되고 변환된 기계어로 컴퓨터가 소스코드를 실행함
* 프로그래밍 언어 종류 : C, C++, C#, JAVA, Python 등
* JAVA는 JVM 컴파일러 사용
* 소스코드로 작성한 .java 파일이 JVM을 통해서 바이트 코드인 .class파일로 변환

1. 자바의 장점

* 컴파일러가 모든 운영체제에서 동일하게 동작함.
* 객체지향 프로그래밍 언어
* 컴파일러인 JVM에 GC(가비지 컬렉터)가 자동으로 메모리 정리

1. 변수(variable)

* 변수는 임의 값을 저장하기 위한 공간
* 변수에는 하나의 값만 저장 가능
* 변수를 사용하기 위해서 변수 선언 필수
* 변수 선언 : 변수의 타입 + 변수의 이름 ex) int value;
* 변수의 이름 명명규칙

첫 번째 글자는 영문자나 $, \_만 가능, 숫자는 불가능

변수의 이름은 대문자, 소문자 구분

변수의 이름은 CamelCase 명명규칙을 따름 ex) int phoneNum;

자바의 예약어들은 변수의 이름으로 지정이 불가능

1. 변수의 사용

* 변수의 값 저장하려면 등호(=) 사용하여 저장
* 프로그래밍 언어에서 등호(=)는 왼쪽 변수에 오른쪽 값을 저장

ex) int value = 1;

* 변수의 값을 지정해주는 것을 변수의 초기화(initialize)라고 함.
* 초기화되지 않은 변수를 사용할 시 에러발생

1. 변수의 타입

* 자바에서 제공하는 기본 타입으로 정수형, 실수형, 논리형 타입이 있음
* 정수형 : int, long, short, char
* 실수형 : float, double
* 논리형 : Boolean

1. 정수형 타입

* int형 정수를 표현하는 대표적인 타입 : 4byte까지 표현 가능
* long형 : 8byte까지 표현 가능
* byte형 : 기본이 되는 정수형 타입, 1byte 표현 가능
* short형 : 2byte 표현 가능’
* char형 : 문자 하나를 표현할 때 사용하는 타입, 2byte 표현 가능

1. String 타입

* 자바에서 기본으로 제공되는 타입이 아니고 자바에서 char 배열을 이용해서 만들어놓은 클래스
* 문자열은 큰 따옴표로 표기
* 문자열 안에 큰 따옴표를 표출할 때는 이스케이프 문자(\) 사용하여 표출
* 자주 사용되는 이스케이프 문자

\” : 큰 따옴표 표출

\n : 줄 바꿈

\t : 탭만큼 띄움

1. 실수형 타입

* float형 : 4byte까지 표현 가능
* double형 : 8byte까지 표현 가능
* JVM에서 실수형 타입은 기본적으로 double형 타입으로 컴파일됨
* float형 변수에는 리터럴 문자인 f를 붙여줘야 됨

1. 논리형 타입

* boolean형 : true, false만 표현 가능

1. 타입변환

* 소스코드를 작성하다보면 타입이 다른 변수들을 연산해야 될 경우가 발생하는데

이 때 허용범위가 작은 타입이 허용범위가 큰 타입으로 자동변환되게 됨

ex) byte num1 = 10;

int num2 = num1; //num1이 int형으로 자동으로 타입변환되어 num2 저장

* 자동 타입변환은 허용범위가 작은 타입에서 허용범위가 큰 타입으로 변환
* 강제 타입변환 : 허용범위가 큰 타입을 허용범위가 작은 타입으로 변환할 때 사용

ex) int intVal = 10;

byte byteVal = (byte)intVal;

* 연산 중 타입변환은 자동 타입변환가 동일하게 허용범위가 큰 타입으로 변환
* + 연산에 문자열 포함된 경우에는 모든 타입의 변수들이 문자열로 변환되어 문자열 결합 연산을 함
* 문자열 강제 타입변환은 자바에서 제공하는 타입별 클래스의 메소드를 이용해서 진행

ex) String str1 = “100”;

int num1 = Integer.parseInt(str1);

1. 시스템 입출력

* 출력하는 방식은 System.out을 이용하여 진행
* System 현재 사용중인 시스템 클래스
* out 시스템에서 사용중인 출력장치
* println() 메소드를 사용하면 모니터로 출력됨
* println() : 괄호안의 내용을 출력 후 줄바꿈
* print() : 괄호안의 내용을 줄바꿈없이 출력
* printf(“형식 문자열”, 값1) : 형식문자열에 맞는 값을 출력

정수형을 표출할 때는 %d라는 형식 문자열

실수형을 표출할 때는 %f라는 형식 문자열

문자열을 표출할 때는 %s라는 형식 문자열을 사용

ex) System.out.printf(“%d”, 10);

System.out.printf(“%d, %f, %s”, 10, 10.1, “string”);

* 시스템 입력하는 방식은 System.in을 이용하여 진행
* in : 시스템에서 사용중인 입력장치
* read() 메소드를 이용하여 입력장치에서 입력된 코드를 저장

1. Scanner 클래스

* System.in.read()의 문제점은 입력된 키 하나하나가 코드로 변경됨
* 한글 같은 자음, 모음 조합을 표현할 수 없음('가', '나', '다'......)
* Scanner 클래스를 이용하면 엔터 값 전까지의 입력값을 문자열로 저장

1. 연산자

* 연산에 사용되는 기호들을 연산자라고 함.

+, -, \*, %,=,+=....

1. 연산자의 종류

* 산술 연산자 : +, -, \*, /, %
* 부호 연산자 : +, -
* 문자열 결합 연산자 : +
* 대입 연산자 : =, +=, -=, \*=, /=, %=
* 증감 연산자 : ++, --
* 비교 연산자 : ==, >, <, >=, <=, instanceof
* 논리 연산자 : !, &, |, &&, ||
* 삼항 조건 연산자 : (조건) ? A : B

1. 조건문

* 조건에 따라 다른 결과를 얻기 위해 사용하는 구문
* if else if else 구문 :

if(조건식1){

내용1

} else if(조건식2) {

내용2

} else {

내용3

}

조건식1이 true일 때 내용1이 실행,

조건식1이 false고 조건식2가 true일 때 내용2 실행

조건식1과 조건식2가 모두 false일 때 내용3이 실행

* switch case 구문

switch(변수) {

case 값1 :

내용1

break;

case 값2 :

내용2;

break;

default :

내용3;

break;

}

변수 값에 따라 실행되는 내용이 달라지는 조건문

변수값이 값1일 때 내용1 실행

변수값이 값1이 아니고 값2일 때 내용2 실행

변수값이 값1과 값2 모두 아닐 때 내용3 실행

1. 반복문

* 특정 소스코드 반복실행 시키고 싶을 때 사용하는 구문을 반복문이라고 함
* for(초기화식; 조건식; 증감식) {

내용1

}

초기화식에 의해 초기화된 변수가 조건식에 충족하면 계속 반복

* while(조건식) {

내용1

}

조건식이 true일 때 내용1이 반복 실행

* do {

내용1

} while(조건식);

내용1이 최초에 실행되고 while 조건식으로 조건검사