1. 자바 시작하기

자바란? 초기의 자바는 가전 제품에 탑재할 프로그래밍 언어로 개발

* 자바의 특징

이식성이 높다

: 서로 다른 실행 환경을 가진 시스템 간에 프로그램을 옮겨 실행할 수 있는 것

객체 지향적이다

: 객체지향언어? 부품에 해당하는 객체들을 먼저 만들고, 이것들읗 하나씩 조립 및 연결해서 전체 프로그램을 완성하는 기법(OOP : Object Oriented Programming)

메모리를 자동으로 관리

: 객체 생성시 자동적으로 메모리 영역을 찾아서 할당하고, 사용이 완료되면 쓰레기 수집기를 실행시켜 자동적으로 사용하지 않는 객체를 제거

멀티스레드를 쉽게 구현

동적 로딩을 지원

1. 변수와 타입

변수 : 하나의 값을 저장할 수 있는 메모리 공간

변수 작성 규칙 : 첫번째 글자는 문자이거나 ‘$’ , ‘\_’ 이어야 하고 숫자로 시작할 수 없다

데이터 타입

|  |  |  |
| --- | --- | --- |
| 값의 종류 | 기본타입 | 메모리크기 |
| 정수 | byte | 1 byte |
| char | 2 byte |
| short | 2 byte |
| Int | 4 byte |
| long | 8 byte |
| 실수 | float | 4 byte |
| double | 8 byte |
| 논리 | boolean | 1byte |

Char 타입

: 자바는 모든 문자를 유니코드로 처리한다. 유니코드는 0~65535범위의 2byte 크기를 가진 정수값이다. 자바는 하나의 유니코드를 저장하기 위해 2byte 크기인 char 타입을 제공한다. 유니코드는 음수가 없기 때문에 char 타입의 변수에는 음수앖을 저장할 수 없다. char타입 변수에 작은 따옴표(‘)로 감싼 문자를 대입하면 해당 문자의 유니코드가 저장된다. Ex) char var1 = ‘A’;

Char변수에 작은 따옴표(‘)로 감싼 문자가 아니라 직접 유니코드 정수값을 저장할 수도 있다. Ex) char c=’A’; int uniCode = c;

Long 타입

: int 타입의 저장범위를 넘어서는 큰 정수는 반드시 정수값 뒤에 l이나 L을 붙여야 한다.

Float 타입

: 자바의 실수 리터럴의 기본타입은 double이기 때문에 float 타입 변수에 저장하려면 리터럴 뒤에 소문자 ‘f’ 나 대문자 ‘F’를 붙여야 한다.

타입 변환

* 자동 타입 변환 : 프로그램 실행 도중에 자동적으로 타입 변환이 일어나는 것

자동타입변환

큰 크기 타입 = 작은 크기 타입

Byte b = 10;

Int i = b; // 자동 타입 변환이 일어난다.

예외) 음수가 저장될 수 있는 byte 타입은 char 타입으로 자동변환시킬 수 없다.

* 강제 타입 변환 : 강제적으로 큰 데이터 타입을 작은 데이터 타입으로 쪼개어서 저장하는 것 ( 캐스팅 : Casting )

강제 타입 변환

작은 크기 타입 = (작은 크기 타입)큰 크기 타입

Int I = 103029770;

Byte b = (byte) I; // 강제 타입 변환 ( 캐스팅 )

* Int 타입은 char 타입으로 자동 변환되지 않기 때문에 강제 타입 변환을 사용해야 한다.

Int I = ‘A’;

Char c = (char) I;

System.out.println(c);

* 실수타입은 정수타입으로 자동 변환되지 않기 때문에 강제 타입 변환을 사용해야 한다. 이 경우 소수점 이하부분은 삭제되고 정수 부분만 저장된다.

Double d = 3.14;

Int I = (int) d; // I 에 3만 저장

1. 연산자

* 연산자 우선순위

1. 최우선 연산자 : () , [] , .
2. 단항 연산자 : !, ~, ++ , -- , (cast)
3. 산술 연산자 : + , - , / , %
4. 쉬프트 연산자 : << , >> , >>>
5. 비교 연산자 : < , > , == , <= , >=
6. 논리 연산자 : & , ^ , && , ||
7. 삼항 연산자 : 조건문 ? 참 : 거짓
8. 대입 연산자 : = , \*= , &=
9. 후위 연산자 : ++ , --
10. 컴마 연산자 : .
11. 조건문과 반복문

(흐름)제어문 = 조건문(if문, switch문) , 반복문(for문, while문, do~while문)

조건문 : 제어문 블록이 실행 완료되었을 때 제어문 블록을 빠져나온다

반복문 : 제어문을 되돌아가 반복실행 한다

Switch문 : 조건에 만족해서 불록 밖으로 빠져나가지 않음. (break 사용)

Switch(변수) { } : 변수 자리에 실수는 사용할 수 없음

Do-while문 : 조건이 만족하지 않아도 최소 한번 실행

Break문 : 대게 if문과 같이 사용되어 if문의 조건식에 따라 for문과 while문을 종료할 때 사용

Public static void main(String[] args) {

While(true) {

Int num = ++1;

If(num==6) {

Break;

}

}

System.out.println(“프로그램종료”);

}

: if문이 참이면 break가 실행되어 while문을 빠져나옴

Continue 문

: 반복문인 for문, while문, do-while문에서만 사용되는데, 블록 내부에서 continue문이 실행되면 증감식, 또는 조건식으로 이동한다.

: continue 문도 대개 if문과 같이 사용 되는데, 특정 조건을 만족하는 경우에 continue문을 실행해서 그 이후의 문장을 실행하지 않고 다음 반복으로 넘어간다.

1. 참조타입

자바의 데이터 타입 : 기본타입 , 참조타입

기본타입 : 정수, 실수, 문자, 논리 리터럴을 저장하는 타입

참조타입 : 객체의 번지를 참조하는 타입 ( 배열, 열거, 클래스, 인터페이스 타입 )

변수 : 스택영역에 저장

객체 : 힙 영역에 저장

메모리 사용 영역

메소드(method) 영역 : 클래스

: JVM이 시작할때 생성되고 모든 스레드가 공유하는 영역이다

힙(heap)영역 : 힙 영역은 객체와 배열이 생성되는 영역

JVM 스택(stack) 영역 : 스택 영역은 각 스레드마다 하나씩 존재하며 스레드가 시작될 때 할당된다.

String 타입

String name1 = ‘윤다솜’;

String name2 = ‘윤다솜’;

: 자바는 문자열 리터럴이 동일하다면 String 객체를 공유하도록 되어 있다. 다음과 같이 name1, name2 변수가 동일한 문자열 리터럴인 “윤다솜”을 참조할 경우 name1과 name2는 동일한 String 객체를 참조하게 된다.

- 배열

: 배열은 같은 타입의 데이터만 저장할 수 있다. 한번 생성된 배열은 길이를 늘리거나 줄일 수 없다.

배열 선언

Int[] arr; int arr[];

: 배열도 객체이므로 힙영역에 생성된다.

Int [] arr = { 0, 1, 2 };

배열 값 수정

Arr[0] = 1;

New 연산자 사용

Int[] arr = new int[] { 1, 2, 3 };

String[] arr = null;

Arr = new String[] {“윤다솜” , “김자바” };

New 연산자로 배열 생성

Int[] arr = new int[5];

배열 길이 얻기

Arr.length;

다차원 배열

Int[][] scores = new int[2][3];

0 1 2

0 (0,0) (0,1) (0,2)

1 (1,0) (1,1) (1,2)

Int[][] arr = { {1,2} , {3,4} };

Int arr = arr[0][0];

1. 클래스

객체(Object) : 물리적으로 존재하거나 추상적으로 생각할 수 있는 것 중에서 자신의 속성을 가지고 다른 것과 식별 가능한 것을 말한다. 객체는 속성과 동작으로 구성되어 있다.

예를들어 사람은 이름, 나이의 속성과 웃다, 걷다등의 동작이 있다. 자바는 이 속성과 동작들을 각각 필드(field)와 메소드(method)라고 부른다.

객체 모델링(Object Modeling) : 현실 세계의 객체를 소프트웨어 객체의 필드와 메소드로 설계하는 것

객체지향 프로그래밍의 특징

1. 캡슐화(Encapsulation) : 객체의 필드, 메소드를 하나로 묶고, 실제 구현 내용을 감추는 것

노출시킬 것인지, 숨길 것인지를 결정하기 위해 접근 제한자를 사용한다.

1. 상속(Inheritance) : 부모 역할의 상위 객체와 자식 역할의 하위 객체가 있는데 상위의 객체는 자기가 가지고 있는 필드와 메소드를 하위 객체에게 물려주어 하위 객체가 사용할 수 있도록 해준다. – 반복된 코드의 중복을 줄여준다.
2. 다형성(Polymorphism) : 다형성은 같은 타입이지만 실행 결과가 다양한 객체를 이용할 수 있는 성질을 말한다. 자바는 다형성을 위해 부모 클래스 또는 인터페이스의 타입 변환을 허용한다. 다형성의 효과로 객체는 부품화가 가능하다.