고건 – Lecture3, 5 정유진 – Lecture4, 6

Lecture 3 – 변수, 타입, 연산자

Java 프로그램의 기본 구조는 우선 소스 파일 안에 클래스를 정의하고 그 안에 메소드가 있고 그 내부에 실행문을 작성하는 형태이다. 중괄호로 각각의 단위를 구분한다.

변수를 선언할 때는 문자나 언더바, $ 로 시작해야 한다. 자바 키워드를 사용하면 안 된다. 자바 키워드에는 int, if, class 등이 있다. 변수를 선언할 때는 먼저 변수의 타입을 지정해줘야 한다. 예를 들어 age는 양의 정수이므로 int age = 20 이라고 할 수 있다. 데이터 타입의 다른 예는 문자에는 char, 실수는 float, double 이 있다. 각각의 타입들이 차지하는 공간의 크기는 각각 다른데, 같은 실수 타입이라도 float은 4 byte, double은 8 byte를 차지한다.

연산자는 데이터 연산을 위한 기호인데, 새로 배우게 된 내용은 증감 연산자이다. 변수 a에서 a++와 ++a는 연산의 순서가 다른데, 전위 증감 연산자인 ++a는 먼저 a의 값을 1 증가시켜 주고 계산한다. a++는 먼저 a의 값을 1 증가하지 않고 기존 값으로 연산을 수행한 뒤, 다음 연산부터 1 증가한 값을 적용한다. 논리 연산자 중 &&는 and, ||는 or을 뜻한다.

변수를 활용하기 위해서는 변수의 생성, 선언의 과정이 필요하다. 이때 오류 방지를 위해 변수를 초기화하는 작업이 필요하다. 예를 들어 int weight = 100; 등으로 선언과 동시에 변수의 값을 지정해줌으로써 초기화할 수 있다. 초기화하지 않은 변수를 연산에 활용하면 에러가 뜨기 때문이다.

데이터를 출력하기 위해서 세 가지의 문법을 활용할 수 있는데 먼저 System.out.println()은 내용 출력 후 행을 바꿔주고, System.out.print()는 내용 출력 후 행을 바꾸지 않고, System.out.printf()는 포맷을 지정하여 출력해준다. 예를 들어 int x = 5; 라고 선언한 뒤 출력할 때 문자열 내부에서 %d라 명시해 준 다음, 뒤에 정수 포맷을 작성하여 대응시켜 줄 수 있다.

Lecture 4

**JAVA 기본 문법**

JAVA에서 정수와 정수끼리 연산하면 그 결과는 정수가 나오고 정수와 실수를 연산하면 그 결과는 실수가 나옵니다. 대입할 값과 변수의 타입이 서로 다른 경우를 타입 불일치라 하는데, 문제가 생길 수 있습니다. 예를 들어 문자열을 정수형 변수에 대입하는 코드는 컴파일되지 않습니다. 이럴 때에는 변수와 대입값을 일치시켜야 합니다.

자동 타입 변환이란 암묵적으로 타입이 바뀌는 것을 말하며 작은 타입의 데이터는 같은 범주의 큰 데이터 타입으로 자동 변환됩니다. 예를 들어 정수는 실수로 자동 변환됩니다. 그 이유는 실수는 정수 뿐 아니라 소수점 이하의 정보도 포함할 수 있기 때문입니다.

강제 타입 변환이란 명시적으로 타입을 강제 변환하는 것을 말하며 타입을 강제로 바꾸기 위해 캐스팅을 사용합니다. 여기에서 캐스팅(Casting)이란, 같은 범수인 데이터의 해석을 달리하는 문법으로, 소괄호 사이에 해석 타입을 명시합니다. 예를 들어, 정수는 실수보다 범위가 작으므로, 10.4와 같은 수는 정수형 변수에 담을 수 없습니다.

캐스팅의 종류에는 다운 캐스팅 (Down-casting)과 업 캐스팅 (Up-casting)이 있는데 다운 캐스팅은 넓은 범위에서 좁은 범위로 해석하는 것을 말하고, 업 캐스팅은 좁은 범위에서 넓은 범위로 해석하는 것을 말합니다.

숫자를 가리키는 문자열끼리 덧셈 연산을 하면 숫자 계산이 아닌 문자열이 연결됩니다. 이를 위해서는 문자열과 숫자(정수&실수) 간 변환을 위한 메소드를 사용합니다.

연산자의 종류에는 비교 연산자, 동등 연산자, 논리 연산자 등이 존재하는데 연산자 간 실행 순서에 유의하여 프로그램을 코딩해야 합니다.

키보드로 데이터를 입력하면 화면에 데이터가 출력되는데, 키보드로 데이터를 입력하면 프로그램의 첫 행에 import java.util.Scanner;를 추가해 Scanner 클래스의 경로 이름을 컴파일러에 알려줍니다. 또 키보드로 데이터를 입력 받기 위해 System.in 객체와 연결된 Scanner 객체를 생성하고 Scanner 클래스가 제공하는 다양한 메서드를 이용해 키보드로 데이터를 입력 받습니다. 예를 들어 int x = in.nextInt();는 정수를 읽어 변수 x에 대입하는 메서드입니다.

**메소드 (Method)**

메소드의 구조는 public static TYPE NAME (PARAMETERS) { ... return VALUE; }이고 TYPE은 변환 타입, NAME은 메소드명, PARAMETERS는 입력 변수, return VALUE;는 변환값을 나타냅니다. 메소드명 (method name)은 메소드의 이름으로, 호출시에 사용하고, 매개변수 (parameters)는 메소드 실행 시 입력값을 저장하는 변수로, 입력변수 혹은 파라미터라고도 부릅니다. 또한 변환값 (return value)은 메소드의 최종 결과로 반환될 값을 의미하고, 반환 타입 (return type)은 반환값의 타입을 의미합니다.

메소드 호출은 메소드명을 적고, 소괄호 사이에 입력값을 전달하여 사용합니다. 메서드를 호출하면 제어가 호출된 메서드(callee)로 넘어갔다가, 호출된 메서드의 실행을 마친 후 호출한 메서드(caller)로 다시 돌아옵니다. 단, return 문을 사용하면 메서드의 실행 도중에도 호출한 메서드로 제어를 넘길 수 있습니다.

매개변수에는 형식 매개변수 (formal parameter)와 실 매개변수 (actual parameter)가 있는데 형식 매개변수는 선언에서 사용된 매개변수를 나타내고, 실 매개변수는 호출에서 사용된 매개변수로 인자(argument)라고도 합니다.

매개변수의 전달이란 호출 시에 실 매개변수와 형식 매개변수를 매칭하는 것을 의미하는데 전달 방식에는 값 전달 (pass by value), 참조 전달 (pass by reference), 값-결과 전달 (pass by value-result), 이름 전달 (pass by name)이 존재합니다.

메소드는 매개변수의 유무로 매개변수 없음, 단일 매개변수, 다중 매개변수로 나누어지고 변환값의 유무로 메소드의 반환값은 존재하지 않는 void 메소드가 있습니다. 이 경우, 반환 타입은 void가 되고, return 키워드는 생략이 가능합니다.

Lecture 5 – 제어문

제어문에는 조건문, 반복문, 분기문 등이 있다. 조건문이란 조건식의 조건을 만족하는 경우와 그렇지 않은 경우를 나누어 실행을 다르게 하도록 만든 문장이다. If문, switch문 등이 있는데 if문에는 if-else문, 다중 if문, 중첩 if문이 있다. if문의 형식은 if (조건식) { 조건식이 참일 때 수행할 내용 } 이다. If-else문은 기존 if문 뒤에 else { 조건식이 거짓일 때 수행할 내용 } 을 추가된 형식이다. 다중 if문은 if문 뒤에 else if (n번째 조건식) { n번째 조건식이 참일 때 수행할 내용 } 을 작성한 뒤 마지막에 else { }를 작성해주면 된다.

switch문은 case에 따라 수행하는 방식을 나누는 문법이다. If-else문을 switch문으로 바꿀 수 있다. switch문의 형식은 switch (기준값) { case “A” : 기준값 = A일 때 수행할 문장 } 이다. Default 는 case에 없는 나머지 경우일 때 수행할 문장을 입력하면 된다. Java14부터는 case “A” -> 실행문의 형식으로 간단히 구현할 수 있게 되었다.

Lecture 6

**조건문 (cont.)**

삼항 연산자는 조건식에 따른 반환값을 지정하는 문법입니다. 조건 식이 참이면 (조건식) ? A : B;에서 A, 거짓이라면 B를 반환합니다.

**반복문**

반복문(loop)이란 규칙적 반복 코드를 단순화하는 문법을 말합니다. 즉, 조건에 따라 같은 처리를 반복하게 하는 문법입니다. 수천 또는 그 이상의 코드를 단 몇 줄로 줄일 수 있는 장점이 있습니다.

반복문은 크게 while 문과 for 문으로 구분하는데 while 문은 반복 횟수가 명확하지 않을 때 사용하기 적합하고 for 문은 반복 횟수가 명확할 때 사용하기에 적합합니다.

무한 루프(Infinite Loop)란 반복이 끊임없이 계속되는 상황을 말하는데 반복문 사용시 무한 루프에 빠지지 않도록 주의해야 합니다. 실행 흐름이 반복문에 갇히면 프로그램을 멈출 수 없기 때문입니다.

while 문은 반복 횟수가 조건에 따라 변할 때 사용하기 좋은 반복문이고 while (조건식) { 반복내용 }에서 조건식이 참인 경우, 중괄호 안쪽 코드를 반복합니다. 해당 과정은 조건 식이 거짓이 되어야만 반복을 벗어날 수 있습니다. 예외로 do-while 문은 조건식이 거짓이라도 한 번은 본체를 실행해 줍니다.

for 문은 반복 횟수가 명확할 때 사용하기 편리한 반복문이고 for (초기화; 조건식; 갱신) { 반복 내용 }에서 초기화는 단 한 번 수행되며, 그 이후 조건식 검사, 반복 내용 수행, 갱신이 순차적으로 진행됩니다. 조건식이 참이라면 해당 과정은 계속해서 반복됩니다.

**분기문**

break 문은 반복문을 탈출하는 문법이고 continue 문은 실행 흐름을 다음 반복으로 넘기는 문법입니다. 즉, 반복문의 모든 코드를 수행하지 않은 채 다음 반복으로 이동합니다.