KH 1주차 정리

1. 변수를 만드는 방법

   Int(데이터 타입) studentNum(변수 명) = 0(초기화 값);

   (1) 데이터 타입을 적는다.

   (2) 변수 이름을 작성한다. (우리가 만들고 싶은 대로 지정 가능)

   (3) 필요하다면 초기화를 진행한다. (변수에 값 대입)

➔ 결론적으로 변수를 만들 때 가장 중요한 것은 변수이름이다. 변수이름이 무엇을 의미하는 것인지 명확하게 전달하기 위한 목적이 가장 중요하다.

2. for문을 만드는 방법

   for(초기화; 조건; 증감) { 조건식이 만족하면 반복할 코드; }

   (1) for를 작성하고 소괄호( ) 작성 후 중괄호{ }를 작성한다.

   (2) 소괄호 내부는 위의 명시한 것과 같다.

   (3) 중괄호 내에는 for문을 반복하며 작업할 내용을 적는다.

➔ 중요한 것은 for 문은 조건 파트가 참인 동안은 언제든지 반복된다는 것이다.

3. if문을 만드는 방법

   if (조건식) { 조건식이 만족하면 실행되는 코드; }

   else if (조건식) { 위의 if문이 아니고, 현재 조건식이 만족하면 실행되는 코드; }

   else { 위 if문 들의 조건식이 아니라면 실행하는 코드; }

   (1) if를 작성하고 소괄호( ) 작성하고 중괄호{ }를 작성한다.

   (2) 소괄호 내부에 조건식을 작성한다.

   (3) 중괄호 내부에는 조건이 만족된 경우 동작할 코드를 작성한다.

➔ 만약 if, else if, else if 형태로 코드가 작성되면 depth(깊이)가 길어질수록 코드를 파악하기 위한 혼동이 가중된다는 문제가 있다. 이와 같은 이유로 코드를 작성할 때, 그냥 if, if, if가 더 좋다. (전제조건 : 서비스 개발자)

4. Switch 문을 작성하는 방법

Switch (조건) { case 0: 0이면 실행할 코드; break; …

default: 위의 조건이 만족하지 않으면 실행되는 코드; break; }

(1) Switch를 적고 소괄호( )를 작성하고 중괄호{ }를 작성한다.

(2) 소괄호 내부에 switch case에서 사용할 조건을 적는다.

(3) 중괄호 내부에는 case 조건들을 적고 각 조건에 대응하는 코드를 작성하면 된다.

5. While 문을 작성하는 방법

While (조건식) { 조건식이 참이라면 실행되는 코드; }

(1) While을 적고 소괄호( )를 작성하고 중괄호{ }를 작성한다.

(2) 소괄호 내부에 조건식을 작성한다.

(3) 중괄호 내부에 조건이 만족되는 동안 반복시킬 코드를 작성한다.

6. Final을 사용하는 이유

(1) 상수를 고정시킬 수 있다는 이점이 있다. (★유지보수)

(2) 불변 객체이다. (★사고방지)

7. Scanner

Scanner scan = new Scanner(System.in);

Scanner는 키보드 입력 처리를 위해 사용하는 객체이다.

만약 키보드 입력으로 int 타입을 원한다면 **nextInt( )**를 사용한다.

Int inputNumber = scan.nextInt( );

8. 랜덤 숫자 만드는 방법

Int dice = (int)(Math.random() * MAX) + MIN;

(1) Math.random() 을 작성한다.

(2) 최소값과 최대값을 확인한다.

(3) 최소값은 더하기로 표기해주고, 최대값은 곱하기로 표기해준다.

실제 최대값 계산은 (곱하는 값 + 최소값 -1)이다.

9. 배열을 만드는 방법

Int[ ] numberArray = {1, 2, 3, 4, 5};

(1) 데이터 타입을 적고 대괄호[ ]를 적는다.

(2) 변수 선언하듯 변수이름을 작성한다.

(3) 필요하다면 중괄호{ }를 열고 초기화를 하거나 new 데이터 타입[ ] 형태로 Heap 메모리에 강제 할당할 수 있다.

10. Heap과 Stack

Stack은 Loop { final int data } 형태가 있다면 data 변수가 루프마다 초기화되는 것을 볼 수 있다. 이런 지역변수 특성을 가지는 것들이 Stack이다. 반면 new를 해서 Heap에 설정되는 정보들은 메모리에 상주하게 된다. 그러므로 언제 어디서든 데이터에 접근할 수 있게 된다.

➔ 결론 : Stack은 중괄호{ } 내에서 사용되고, Heap은 new하고 이후로 사용된다.

11. For의 변형 버전, foreach 사용법

For (int num : numberArray) { 실행할 반복 문 }

(1) 배열의 데이터 타입을 작성한다.

(2) 배열의 원소를 표현할 이름을 적당히 지정한다.

(3) 콜론 하나 찍는다 (:)

(4) 정보를 하나씩 꺼내 올 배열을 적어준다.

12. OOP (Object Oriented Programming), Domain Driven Development (DDD)

OOP란 모든 정보를 객체 화하여 레고처럼 필요하면 조립하여 관리하자라는 뜻을 가진다. Domain이라는 관점은 이렇게 클래스가 어떤 주제에 집중을 하고 있는 지를 본다고 생각하면 된다. 즉, 내가 집중하는 주제가 무엇인가를 알 수 있도록 예쁘게 잘 표현해주는 것을 OOP라고 보아도 무방하다.

13. Class

class는 필드, 생성자, 메서드로 구성되어 있으며, 필드는 변수 선언, 생성자는 상태, 메서드는 객체의 활동을 만들어준다.

13-1. 생성자 : 생성자는 class의 이름과 같으며, 리턴 타입이 없다.

(1) public을 적고 클래스 이름을 적은 이후 소괄호()를 작성 후 중괄호{}를 작성한다.

(2) 만약에 외부에서 값을 입력 받을 것이라면 소괄호에 입력 받을 형태를 작성한다.

(3) 실제 클래스가 new를 통해 객체화 될 때 구동 시키고 싶은 작업을 중괄호 내부에 배치한다.

13-2. 메서드 : 클래스 내부에 기능을 수행하는 집합들을 메서드라고 한다.

(1) public을 작성하고 소괄호()를 작성하고 중괄호{}를 작성한다.

(2) 리턴 타입을 public 옆에 작성한다.

(3) 메서드의 이름을 그 옆에 작성해준다.

(4) 중괄호 내에서는 실제 메서드 이름에 해당하는 작업을 진행하면 된다.

13-3. Getter / Setter

Getter는 class 내에서 다루는 정보를 얻기 위해 사용하고, Setter는 class 내에서 다루는 정보를 직접 설정하는 목적으로 사용한다.

14. 리팩토링

외부에서 보는 프로그램 동작은 바꾸지 않고 프로그램 내부 구조를 개선하는 것을 의미한다. 외부 프로그램을 사용하는 사용자일 수도 있고, 클래스나 메서드를 사용하는 다른 클래스나 메서드 일 수 있다.