1주차 정리

1. 변수 - 데이터타입 변수이름 = 초기화값;

1) 데이터 타입을 작성한다 - int, float, double, long 등 (Class형 데이터 타입도 가능)

2) 변수 이름을 작성한다. - 변수가 무엇을 의미하는지 명확하게 전달하기 위해
변수 이름을 잘 작성해야한다.

3) 필요시 초기화를 진행한다.

** 상수로 고정 시키는 final의 이점

1) 코드 수정시 상수값만 변경하면 일괄처리가 가능해진다.

2) 새로운 값 대입을 못하게 막음으로써 무결성을 보장한다.

2. 제어문

1) if문 - if (조건식) { 작업할 내용 }

- else if(조건식) { 작업할 내용 } → 위의 if와 else if 조건식을 만족되지 않았을 때 현재 조건식 체크를 하게 됨
- else { 작업할 내용 } → 위의 if와 else if 조건식들을 전부 만족되지 않았을 때 동작

** if - else if - else if - else if 식으로 코드가 길어지면 코드 파악에 혼동이 올 수 있다.
이와 같은 이유로 우리같은 서비스 개발자들은 if - if - if 식으로 코드를 작성하는게 더 좋다.

2) while문 - while(조건식) { 작업할 내용 }

(1) 조건식을 만족하는 동안 반복

↱ for문을 처음 만났을 때 동작

3) for문 - for(초기화; 조건식; 증감) { 작업할 내용 }

(1) 조건식을 만족하는 동안 반복

** for(;;) 작성시 조건이 없으므로 무한 반복

3-1) foreach문 - for(데이터타입 원소이름 : 배열) { 작업할 내용 }

→ for(String s : nameArray) { System.out.println(s); }

(1) 배열에서 정보를 하나씩 꺼내옴

4) switch문 - switch (case 사용 조건)

{ case 0: 작업내용 break; → case 사용조건이 0일 경우
case 1: 작업내용 break; → case 사용조건이 1일 경우
defalt: 작업내용 break; } → 위 조건 모두 만족하지 않을 경우

3. 입력문

1) Scanner - Scanner scanner = new Scanner(System.in)

(1) 키보드 입력 처리를 위해 사용하는 객체

(2) int 타입을 받으려면 scanner.nextInt(), float는 nextFloat(), 한 줄을 받으려면 nextLine() 등의 형태로 사용할 수 있다.

4. 배열

1) Stack

(1) 데이터타입[] 변수이름 = {초기화};

(2) 중괄호{}를 이용하여 사용을 하고 지역변수 특성을 가진다.

2) Heap

(1) 데이터타입[] 변수이름 = new 데이터타입[배열 개수];

(2) new를 통해 Heap 메모리 할당

(3) Heap에 설정 된 데이터들은 메모리에 상주하게 되고 언제 어디서든 데이터에 접근 가능

5. Class

1) OOP(Object Orented Programming)와 DDD(Domain Driven Development)

(1) 모든 정보를 객체화하여 필요할 때 조립해서 관리하는 방식

(2) 하나의 클래스에 모든 정보를 관리하게 될 경우 해당 객체가 어떤 목적을 가지고 있었는지에 대해 알아보기 힘들다.

(3) 이 객체가 어떠한 주제에 대한 객체인지 알아보기 쉽게 표현하는 방식

** 객체 스스로가 하기에 표현이 애매해지는 작업들은 Domain Service로 재배치하여
가독성과 유지보수성 향상을 가져올 수 있다.
이로 인해 비즈니스 관점을 좀 더 명확하게 만들어주는 이점을 가진다.

2) 생성자 - public class이름(외부 입력 받을 시 받을 형태) {작업할 내용}

→ public Dice(int diceNum) { this.diceNum = diceNum; }

(1) 생성자 호출 : class이름 변수이름 = new 생성자;

↳ Dice dice = new Dice();

3) 매서드 - public 리턴타입(외부 입력 받을 시 받을 형태) 매서드이름{작업할 내용}

→ public int getDiceNum(){ return diceNum; }

(1) 리턴타입으로는 데이터타입을 작성하면 되고 만약 정보 반환을 하지 않는다면 void를 작성한다.

(2) getter/setter

- getter는 class 내의 정보를 가져올 때 사용하는 것이고 setter는 class 내의 정보를 설정할 때 쓰는 것이다. 하지만 setter를 사용하게 되면 사용한 의도 파악이나 일관성을 유지하기 힘들기때문에 사용을 지양한다.