1. Class

2. 객체지향 프로그래밍(Object Oriented Programming, OOP)

1. 객체지향 프로그래밍이란 자바의 특징 중 하나로 개발자가 프로그램의 부품이 될 설계도를 작성하고 설계도로 부품들을 만들어서 프로그램을 완성해가는 방식의 프로그래밍
2. 설계도의 역할(클래스, 사용자 정의 타입)을 하고 설계도를 토대로 부품(인스턴스, 객체)을 변수형태로 만든다.

```
//예) NumBaseball(<-이것을 타입이라고 보면 된다.) nb = new
//예) public static class NumBaseball(<-이것이 개발자가 직
    return;
}
```

3. 설계도를 부품으로 만들기 위해서는 인스턴스화(객체화) 작업이 필요하다.
4. 인스턴스화에서는 클래스에 정의되어 있는 속성값들(변수들)을 초기화하고 속성들과 기능들을 메모리에 올려주는 작업이 진행된다.

2. Class

1. 클래스는 부품(객체)를 만들어 주는 설계도의 역할을 하면서
2. 한편으로는 사용자가 직접 정의하는 자료형 타입이라고 볼 수 있다.
3. 클래스는 속성(=필드, 멤버변수, 어트리뷰트, ...)과 기능(메소드, 함수, 펑션)이 2개가 포함된다.
4. 속성이란 클래스내에 선언된 변수들이고, 기능이란 클래스내에 정의된 메소들이다.
5. 세상의 모든 사물들을 클래스로 작성하여 객체로 사용할 수 있다.

```
pulic class Human { //<--따라서Human은 '참조형 변수'의 하나
    //사람의 속성
    int height; //키
    int weight; //몸무게
    int age; //나이
    int skin; //피부색깔

    //사람의 기능
    public void eat() {
        System.out.println("밥을 먹는다");
    }

    //사람의 기능
    public void walk() {
        System.out.println("이족보행한다.");
    }

    //사람의 기능
    public void speak() {
        System.out.println("말을 한다.");
    }
}
```

3. 인스턴스(객체화)와 생성자

1. 인스턴스화

1. 설계도(클래스)를 토대로 부품(객체, 인스턴스)를 만들어 주는 작업을 인스턴스화(객체화)라고 한다.
2. 인스턴스화 작업은 new 키워드와 생성자메소드를 사용하여 진행한다.
3. 인스턴스화 작업은 만들어지는 객체를 메모리에 저장해주는 작업이다. 객체는 참조형 변수이기 때문에 만들어진 객체가 heap메모리에 저장되면서 Class에 정의해 놓은 변수들과 메소드도 메모리에 저장된다. 그러면서 변수들의 초기화가 동시에 진행된다. 기본타입은 0, false로 초기화되고 참조형타입은 null값으로 초기화된다.
4. 위의 작업을 진행하는 메소드가 생성자메소드(생성자)이다.

2. 생성자(생성자메소드)

1. 생성되는 객체를 메모리에 저장하고 객체의 변수들을 초기화해주는 역할을 하는 메소드.
2. 생성자의 형태

```
//이런식으로 생긴게 '생성자의 형태'라고 생각하면 된다.
//특징: 리턴타입이 없고 클래스명과 동일한 이름을 갖는 메소드
public 클래스명() {

}

//NumBaseball파일 참고
```

3. 생성자는 '매개변수가 있는 생성자'와 '매개변수가 없는 생성자'로 구분된다.
4. 매개변수 없는 생성자를 : '기본 생성자'라고 하고, 클래스를 정의하면 생성자를 따로 정의하지 않아도 제공되는 생성자.
5. 매개변수가 있는 생성자

```
public 클래스명(매개변수) {
    //받아온 매개변수들을 통해 속성값의 초기화 가능
    //매개변수에 제한은 없다.
}
```

6. 매개변수가 있는 생성자를 사용하려면 항상 클래스내에 정의를 해야한다.
7. 주의할 점 : 매개변수가 있는 생성자를 정의하면 기본 생성자가 제공되지 않는다. 때문에 기본 생성자도 함께 정의해주는 것이 좋다.