2023-03-14

변수와 메서드

지역변수 : 변수 선언문 수행시

멤버변수 - 클래식변수(static변수, 공유변수) : 클래스가 메모리에 올라갈 때

- 인스턴스변수(nonstatic변수) : 인스턴스 생성시

```
class Variables {
  int iv; // 인스턴스변수
  static int cv; // 클래스변수

  void method() {
    int lv = 0; // 지역변수
  }
}
```

인스턴스변수는 인스턴스가 생성될 때마다 생성되므로 인스턴스마다 각기 다른값을 유지할 수 있지만, 클래스변수는 모든 인스턴스가 하나의 저장공간을 공유하므로 항상 공통된 값을 갖는다.

```
class Card {
  // 인스턴스 변수(객체 생성시마다 각각의 정보를 보관해야하기 때문)
  String kind; // 무늬
  int number; // 숫자

  // 클래스 변수 (고정된 값이라 각각 메모리를 저장할 필요가 없다)
  static int width = 100; // 폭
  static int height = 250; // 높이
}
```

클래스메서드와 인스턴스메서드

인스턴스메서드

- 인스턴스 생성 후, '참조변수/메서드이름()'으로 호출
- 인스턴스변수나 인스턴스메서드와 관련된 작업을 하는 메서드
- 메서드 내에서 인스턴스변수 사용가능

클래스메서드

- 객체생성없이 '클래스이름/메서드이름()'으로 호출
- 인스턴스변수나 인스턴스메서드와 관련없는 작업을 하는 메서드
- 메서드 내에서 인스턴스변수 사용불가
- 메서드 내에서 인스턴스변수를 사용하지 않는다면 static을 붙이는 것을 고려한다.

변수 초기화

변수를 선언하고 값을 저장하는 것

```
int i = 10; // int형 변수 i를 선언하고 10으로 초기화
int i = 10, j = 10; // 같은 타입의 변수는 ,로 함께 선언 혹은 초기화 가능
int i = 10, long j = 0; // 타입이 다른 변수는 함께 선언하거나 초기화 불가.
int i = 10; int j = i; // 변수 i에 저장된 값으로 변수 j를 초기화
```

```
// 명시적 초기화
class Car {
   int door = 4; // 기본형(primitive type)변수의 초기화
  Enginie e = new Engine(); // 참조형(reference type)변수의 초기화
}

// 생성자
Car(String color, String gearType, int door) {
  this.color = color;
  this.gearType = gearType;
  this.door = door;
}

// 초기화 블럭
class InitBlock {
  // - 클래스 초기화 블럭 : static { }
  static { /* 클래스 초기화 블럭 */}
  { /* 인스턴스 초기화 블럭 */ }
}
```

클래스 초기화블럭(static 초기화블럭) : 클래스변수의 초기화가 복잡할 때 사용되며 클래스가 로딩될 때 실행

인스턴스 초기화블럭 : 인스턴스가 생성될 때마다 생성자보다 먼저 실행

💡 **main method가 호출되기 전에 static초기화 블럭이 먼저 실행된다.**

상속

- Java는 기본적으로 단일상속 지원
- 새로운 클래스 생성시 부모 클래스를 1개 지정해야하나 생략하면 java.lang.Object 클래스 상속
- java.lang.Object 클래스는 자바로 만들어지는 모든 클래스들의 최상위 클래스
- 어떤 클래스든 객체를 생성하게되면 조상 클래스의 객체가 함께 생성된다
- 객체 생성시 호출되는 생성자메서드는 호출되자마자 부모클래스의 생성자를 호출하는 특징을 가지고있다.
    - 호출되는 부모클래스의 생성자는 아규먼트가 없는 생성자이다.
- 부모클래스에 아규먼트를 받지않는 생성자 없는 경우엔 **super()** 메서드를 이용해서 부모가 가지고있는 생성자를 직접 호출해야한다.
    - 아규먼트가 없는 경우 컴파일러가 자동으로 super()를 넣어줌으로 직접입력하지 않아도된다.
    - this 가 있는 경우 역시 super()를 사용하지 않아도 된다.

```
class A{
  A() {
    System.out.println("A클래스의 객체 생성!");
  }
}
class B extends A{
  B(int no) {
    System.out.println("B클래스의 객체 생성!");
  }
}
class C extends B{
  C() {
    super(10);
    System.out.println("C클래스의 객체 생성!");
  }
}
```

모든 생성자는 생성되자마자 부모생성자를 호출하는 기능을 갖는다.