Method Overloading

Method Overloading

- 원칙적으로 하나의 Class 안에 동일한 이름의 Method가 두 개 이상 선언될 수 없다.

- 하지만 이름이 같더라도, **Parameter의 개수 또는 자료형 중 하나라도 일치하지 않으면 동일한 이름의 Method를 선언할 수 있는 기능이 Method Overloading**
- **Java Compiler는 Overloading 된 Method 호출 시 전달하는 인자를 통해서 Method를 구분한다.**

```
class AAA {
	// Method Overloading
	void isYourFunc(int n){};
	void isYourFunc(int n1, int n2){};
	void isYourFunc(int n1, double n2){};
}

class MethodOverloading {
	AAA inst = new AAA();
	inst.isYourFunc(10);
	inst.isYourFunc(10, 20);
	inst.isYourFunc(10, 3.15);
}
```

기막히게 애매한 상황?

```
class AAA {
	// Method Overloading
	void isYourFunc(int n){};
	void isYourFunc(int n1, int n2){};
	void isYourFunc(int n1, double n2){};
}

class MethodOverloading {
	AAA inst = new AAA();
	inst.isYourFunc(10, 'a');

	/*
		자동 형변환 규칙이 적용되어 char → int or double로 형변환이 가능한데, 형변환 규칙에 따라
		int로 형변환 된다. 그러나 형변환까지 되는 코드는 만들지 말자.
	*/
}
```

Method Overloading - Constructor

! **생성자도 Overloading이 가능하고, 하나의 Class를 기반으로 다양한 Instance를 만들 수 있다.**
(Instance 생성 방법을 다양하게 만들 수 있다.)

```
class Person {
	private int perID = 0;
	private int milID = 0;

	// Instance 생성 방법 1
	public Person(int perID, int milID){
		this.perID = perID;
		this.milID = milID;
	}

	// Instance 생성 방법 2
	public Person(int pID){
		this.perID = perID;
	}

	public void showInfo() {
```

```
            System.out.println("민번 : "+perID);

            if(milID != 0)
                  System.out.println("군번 : "+milID);
            else
                  System.out.println("군과 관계 없음");
      }

}

Class ConstructorOverloading {
      public static void main(Stirng[] args) {
            Person man = new Person(950123, 880102);
            Person woman = new Person(941125);

            man.showInfo();
            woman.showInfo();
      }
}
```

Keyword this를 통한 생성자 호출

Keyword this를 이용하면 생성자 내에서 다른 생성자를 호출할 수 있다.

this의 두가지 의미

i. **생성된 Instance 자신을 의미**
ii. **Overloading의 경우 다른 생성자를 지칭**

```
class Person {
      private int perID = 0;
      private int milID = 0;
      private int birthYear = 0;
      private int birthMonth = 0;
      private int birthDay = 0;

      public Person(int perID, int milID, int bYear, int bMonth, int bDay) {
            this.perID = perID; // this : Instance 자신을 의미
            this.milID = milID;
            birthYear = bYear;
            birthMonth = bMonth;
            birthDay = bDay;
      }

      public Person(int pID, int bYear, int bMonth, int bDay) {

            this(pID, 0, bYear, bMonth, bDay);  // this : Overloading된 다른 생성자의 호출

      }
}
```

- this가 없을 경우에는 두번째 생성자에 첫번째 생성자의 초기화 문장이 다 삽입되어야 되서 문장의 중복이 발생!
  생성자마다 중복되는 초기화 과정의 중복을 피할 수 있다.