JAVA 방중교육

3. 객체와 클래스(1)

방중교육 진행자: 김준성

TO-DO LIST:

- ☑ 메소드 활용
- ☑ 객체의 소멸
- ☑ 접근 지정자
- ☑ static
- ☑ final

메소드 활용

```
public int increase(int n, int p) {
```

접근 지정자 리턴 타입 메소드 이름 메소드 인자

```
        return n + p;

}
```

...

```
System.out.println(calculator.increase(3, 5));
```

메소드 활용

인자 전달법에는 call by value와 call by reference가 있습니다.

call by value: 기본 타입 값이 전달되는 경우로, 매개변수에 값을 복사시킵니다.

call by reference: 레퍼런스 변수들의 레퍼런스 값이 전달된다.
레퍼런스 변수들에 대한 값이 전달되지 변수 그 자체가 통째로 복사되지는 않는다.

메소드 활용

메소드 오버로딩: 메소드의 이름이 같지만, 변수의 타입이나 개수를 다르게 하여 메소드를 중복 작성하는 방법.
메소드의 리턴 타입이나 접근 지정자를 바꾸는 것은 오버로딩이 되지않습니다.

```
class Calculator {

        public int increase(int i) {

            return i + 1;

        }

        public int increase(int i, int j) {

            return i + j;

        }

}
```

메소드 활용

오버로딩된 메소드 호출은 매개 변수의 타입을 맞추고, 개수를 맞추면 됩니다.

```
Calculator c = new Calculator();

c.increase(1);

c.increase(3, 7);
```

객체의 소멸

자바에는 C++과 달리, 객체를 소멸시키는 연산자가 존재하지 않아, 개발자 마음대로 객체를 소멸시킬 수 없습니다.

Garbage는 더 이상 쓸 모 없어진 객체나 배열의 메모리 입니다.
자바는 Garbage collector가 있어서 알아서 쓸 모 없어진 메모리를 수거해줍니다.

```
Calculator c, d;
c = new Calculator();
d = new Calculator();
c = d;
```

기존의 d에 할당된 메모리는 더 이상 호출될 수 없습니다.
따라서 가비지 콜렉터가 메모리를 수거해갑니다.

객체의 소멸

가비지 컬렉터는 보통 시스템에서 알아서 실행되서 쓸 모 없어진 메모리를 수거해갑니다.

만약 가비지 컬렉션을 강제로 요청하고 싶으면

System.gc(); 명령을 입력하면 됩니다.

접근 지정자

지정자 이름	설명
public	모든 클래스들이 접근 가능하다.
protected	동일한 패키지의 클래스와 자식 클래스들만 접근할 수 있습니다.
default	동일한 패키지 안에 있는 클래스들만 디폴트 멤버를 자유롭게 접근할 수 있다.
private	클래스 자기자신 안에서만 사용가능

static

static은 클래스의 멤버들(필드, 메소드)에 선언할 수 있습니다.
static은 다음과 같은 특징을 가지고 있습니다.

1. static 멤버는 클래스당 하나가 생성됩니다.
클래스에 static 멤버를 1번만 선언할 수 있다는 것이 아니라,
여러 객체가 static 멤버 하나를 공유한다는 뜻입니다.

2. 클래스 로딩 시에 멤버가 생성됩니다.
객체가 new를 통해 생기기 전에도 이미 생성되어 있기에
객체가 생기기 전에도 사용할 수 있고, 객체가 소멸되도 멤버는
사라지지 않습니다.

static

3. static 메소드는 this를 사용할 수 없습니다.
static 메소드는 객체없이도 존재하기 때문에, this를 사용할 수 없습니다.

```
class Calculator {

        static int even = 2;
        public static increase() {...}

}
```

final

final 은 클래스, 메소드, 필드 각각 쓰이는 용도가 다릅니다.

final 클래스: 클래스를 상속할 수 없습니다.

final class Test { ... }

final 메소드: 오버라이딩(오버로딩 아님)을 할 수 없습니다.

protected final void test() { ... }

final 필드: 상수입니다.

final int PI: 3.1415926535