Ch6
클래스

객체지향프로그래밍(기본) 2019

경상대학교 항공우주및소프트웨어공학전공

6.1 객체지향

객체(object)란 사물이나 그에 대한 개념을 의미한다. 집, 사람, 자동차이 모든 것이 객체이다.

객체지향이 프로그래밍이란 프로그램의 기본단위를 변수와 메서드로 이뤄진 객체로 만들어 객체들의 상호작용을 이용한 프로그램을 작성하는 것을 말한다.

객체 내의 특성은 변수에 설정하며 객체의 동작은 메서드를 이용한다.

객체지향의 요소로는 캡슐화, 상속성, 다형성이 있다.

6.2 클래스와 객체

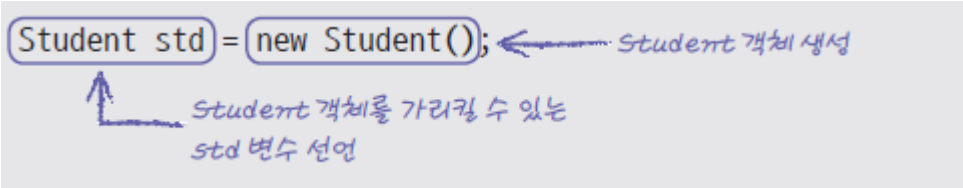
자바는 프로그램을 사용하기 위하여 클래스(class)를 사용한다. 클래스는 객체를 생성하기 위한 틀이다. 클래스를 설계도로 볼 수 있고 객체는 설계도를 가지고 만들어진 제품으로 생각 할 수 있다.

클래스는 다음과 같이 class 키워드를 이용하여 생성할 수 있다.

```
class 클래스이름{
        }
```

클래스와 객체

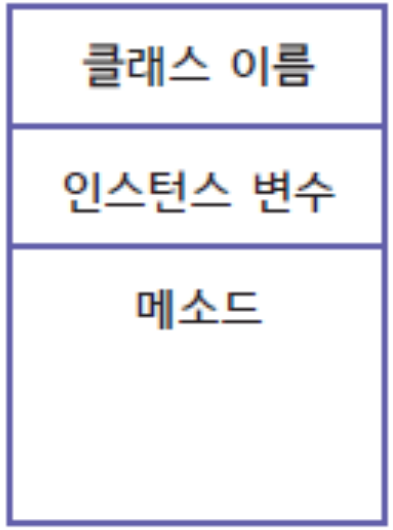
생성된 클래스를 기반으로 new 키워드를 이용하여 객체를 생성한다.



new Student() 가 의미 하는 것을 Student() 로 새로운 Student 클래스의 객체(인스턴스)를 생성하라는 것이다.
= 대입 연산자를 이용하여 std 변수가 Student 클래스의 객체를 가리킨다.

6.3 클래스의 구성요소

클래스는 속성을 나타내는 변수와 행동을 나타내는 메소드로 이루어진다. 변수는 객체가 아는 것 메소드는 객체가 하는 것을 나타낸다. 클래스의 변수는 인스턴스 변수라 한다.



인스턴스 변수만 가진 클래스

인스턴스변수가 값을 넣지 않더라도 객체에 선언된 인스턴스변수는 사용 가능하며 변수가 정수 데이터 타입의 경우 0, 실수 데이터 타입은 0.0, 배열이나 문자열일 경우 null, 부울의 경우 false로 초기화가 되어있다.

```
1  class Student {
2    String name;
3    int id ;
4  }
5
6  public class StudentTest {
7    public static void main(String[] args) {
8      Student std = new Student();
9
10     System.out.println("이름: " + std.name + " 학번: "+std.id);
11     }
12 }
```

Student 객체를 생성하여 std 레퍼런스 변수에 연결

결과

```
이름: null 학번: 0
```

외부에서 인스턴스 변수 접근

```
1  class Student {
2    String name;
3    int id ;
4  }
5
6  public class StudentTest {
7    public static void main(String[] args) {
8      Student std = new Student();
9
10     std.id = 12345;
11     std.name = "Kim";
12
13     System.out.println("이름: " + std.name + " 학번: "+std.id);
14     }
15 }
```

결과

```
이름: Kim 학번: 12345
```

메소드를 이용한 인스턴스 변수 접근

인스턴스변수에 값을 대입하는 메소드

```
void setName(String str)   <-- name변수의 값을 저장하는 메소드
{
  name = str;
}
void setId(int number)   <-- id 변수의 값을 저장하는 메소드
{
  id = number;
}
```

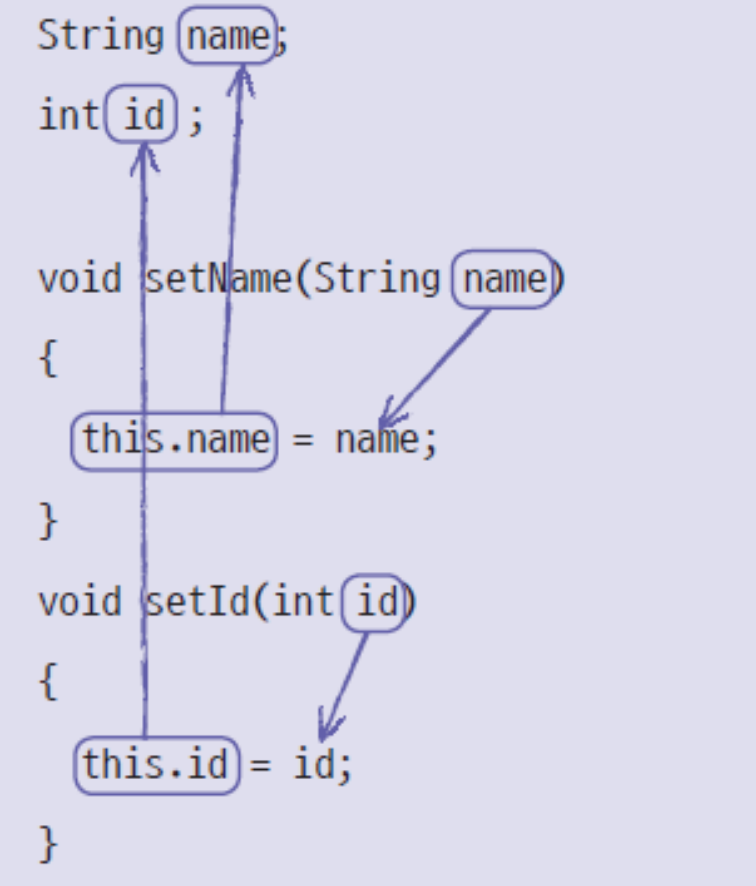
인스턴스변수의 값을 가져오는 메소드

```
String getName()
{
  return name;
}

  int getId()
{
  return id;
}
```

this 키워드

자바에서 클래스의 인스턴스 변수나 메소드를 호출하기 위해 this 키워드를 사용한다. this 키워드는 선언된 클래스 자신을 가리킨다.



6.4 접근 제어지시자

클래스는 인스턴스 변수의 접근을 클래스 내부에서만 할 수 있도록 하는 것을 권장 한다.

캡슐화의 요소를 맞추기 위하여 자바는 접근 제어지시자를 사용한다.

접근 제어지시자는 접근 할 수 있는 권한을 명시하는 키워드로 자바에서 private, default, protected, public을 사용한다.

private 접근 제어 지시자

private 키워드로 선언된 변수나 메소드는 해당 클래스에서만 접근이 가능하며 외부에서 접근을 시도할 시 프로그램은 동작 하지 않는다.

인스턴스 변수는 특별한 경우를 제외하고 private 선언을 하여 사용하는 것을 권장한다.

```
private String name;
private int id ;
```

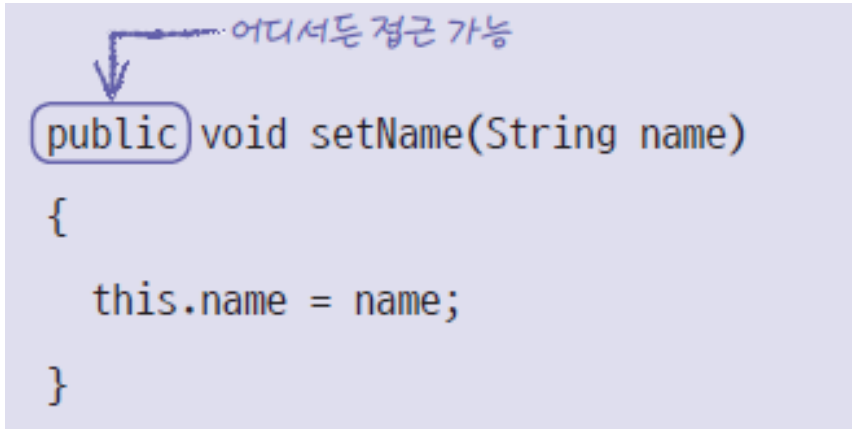
```
System.out.println("이름: " + std.name + " 학번: "+std.id);
```

```
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
    The field Student.name is not visible
    The field Student.id is not visible
```

public 접근 제어 지시자

public 키워드로 선언된 변수나 메소드는 어느 클래스에서라도 접근이 가능하다.

인스턴스 변수는 private로 선언하고 메소드는 public 선언하여 인스턴스 변수의 값의 입력을 메소드를 통해 전달받아 규칙에 맞지 않는 값이 입력 될 경우 그에 대한 처리를 할 수 있다.




```
std.setName("Kim");
```

6.5 생성자

생성자는 객체를 생성 할 때 사용하며 메소드와 비슷하다. 생성자의 이름은 클래스명과 동일하며 반환형을 정의하지 않는다.

new 키워드로 객체가 생성될 때 생성자는 호출 된다.

```
class Student {
  private String name;
  private int id ;

  Student()   <-- 생성자의 선언
  {
  }
```

생성자 호출

```
Student std = new Student();
```

생성자를 이용한 객체의 초기화

생성자는 메소드처럼 매개변수를 사용하여 객체를 생성할 수 있다.

```
Student(String name, int id)
{
  this.name = name;

  if(id<0)
  {
    System.out.println("학번에 음수는 입력 할 수 없습니다.");
  }
  else
  {
    this.id = id;
  }
}
```

```
Student std = new Student("Kim", 12345);
```

생성자 오버로드

생성자는 메소드처럼 오버로드가 가능하다. 생성자의 매개변수의 형태에 따라 동일한 이름을 가진 생성자를 만들 수 있다.

```
Student()
{
  System.out.println("Student() 생성자 호출");

  this.name = "Park";
  this.id = 123;
}
```

```
Student(String name)
{
  System.out.println("Student(String name) 생성자 호출");

  this.name = name;
  this.id = 1234;
}
```

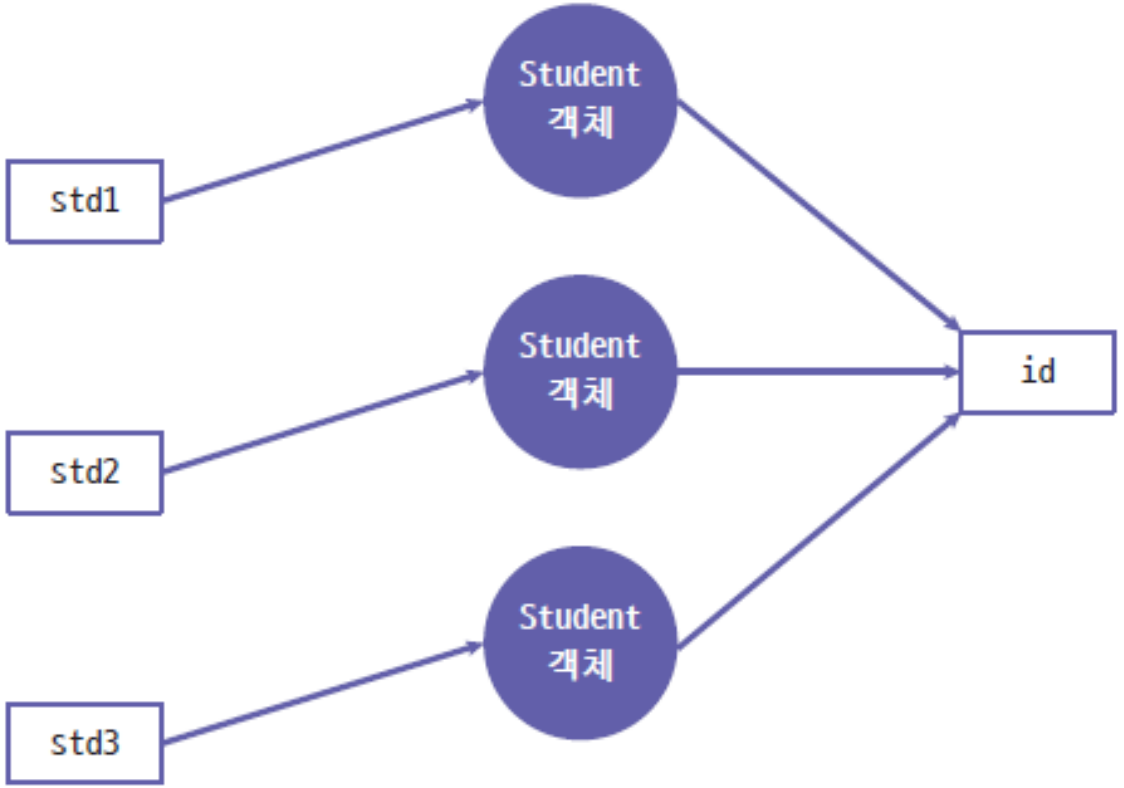
```
Student std1 = new Student();
Student std2 = new Student("Lee");
```

6.6 static 인스턴스 변수

클래스에 static 인스턴스 변수가 선언되면 클래스를 통해 생성된 객체는 모두 같은 변수를 공유하게 된다.

```
class Student {
  private String name;
  private static int id ;
```

static 키워드 선언



6.7 내부 클래스

내부 클래스란 클래스 안에 다른 클래스가 선언 되어 있는 것으로 클래스 안에 선언된 클래스를 내부 클래스라 한다. 내부 클래스는 클래스 안에 선언되어 있기 때문에 클래스의 인스턴스 변수와 메소드가 private로 선언되어 있어도 접근이 가능하다.

```
public class StudentTest {
  private String name;
  private int id ;

  class Student {
    Student(String str, int number)
    {
     }
   }
   public static void main(String[] args) {
     StudentTest stdTest = new StudentTest();
     StudentTest.Student std = stdTest.new Student("Kim", 12345);

     std.studentPrint();
     }
}
```

내부 클래스 객체 생성