고건

Lecture 9

클래스를 선언하려면 class ‘클래스이름’ { } 로 선언한다. 클래스 내부는 크게 필드와 메소드로 나눌 수 있는데, 필드는 객체의 속성을 나타내고, 메소드는 객체의 동작을 나타낸다. 클래스는 보통 하나의 파일에 한 개만 선언하지만, 둘 이상 선언할 수도 있다. 클래스가 둘 이상 있을 때 하나의 클래스만 public으로 선언할 수 있고, 그 클래스 이름은 해당 파일의 이름과 동일해야 한다. 객체를 생성할 때, 먼저 클래스이름 변수; 로 선언해주고 변수 = new 클래스이름() 이렇게 선언할 수 있다. 변수를 생략할 땐 new 클래스이름(); 이렇게 할 수 있다.

클래스의 구성 요소는 멤버와 생성자로 나눌 수 있다. 멤버는 필드와 메소드를 합쳐서 부르는 것이다. 필드와 지역 변수의 차이점은 지역 변수는 매개변수로, 메소드 내부에 선언된 변수이므로 기본 값이 있는 필드와 달리 기본 값을 가지고 있지 않으므로 반드시 초기화를 해야 한다. 필드는 클래스 내부에서 어디서든 사용할 수 있지만, 지역 변수는 선언된 블록 내부에서 선언 후에만 사용 가능하다.

테이블이(가) 표시된 사진

자동 생성된 설명

클래스의 멤버에 접근하기 위해서 해당 클래스 내부에서는 변수 이름 혹은 this.변수이름으로 선언하면 된다. 메소드는 해당 메소드 이름 () 으로 접근하면 된다. 해당 클래스 밖에서 접근하기 위해서는 해당 클래스이름.변수이름 혹은 해당 클래스이름.메소드() 로 접근할 수 있다.

접근 제한자라는 것이 있는데, 이는 해당 클래스의 허용 범위를 정해주는 것이다. 먼저 public은 모든 클래스의 접근을 허용한다. Protected는 동일 패키지 또는 상속 관계의 클래스의 접근을 허용한다. Default는 접근 제한자를 생략했을 때의 기본 값으로, 동일 패키지에 존재하는 클래스의 접근을 허용한다. Private은 모든 클래스의 접근을 허용하지 않는다. private으로 선언된 클래스의 멤버에 접근하려면 접근자와 설정자가 필요한데, 접근자는 값을 반환해주는 것으로 private 필드를 외부로 반환하는 public 메소드이다. Get + 필드명의 형태로 작성하고, 반환 타입은 대상 필드와 같다. 접근자를 public 으로 선언하게 되면 클래스 밖의 모든 코드에서 접근 가능하다. 설정자는 private 필드의 값을 외부에서도 변경 가능하게 하는 public 메소드이다. Set + 필드명의 형태이고, 반환 타입은 void이다. 생성자는 객체를 생성하는 시점에서 필드를 초기화 시켜주는 역할을 한다. 생성자의 이름은 클래스 이름과 같고, 반환 타입은 없다. 생성자는 new 연산자와 함께 사용하며, 객체를 생성할 때 호출하고 오버로딩이 가능하다. 사용자가 생성자를 선언하지 않으면 컴파일러에서 자동으로 디폴트 생성자를 추가해준다. 생성자 역시 메소드와 같이 오버로딩이 가능하다. 클래스의 필드를 static으로 선언하면 모든 객체가 그 필드를 공유할 수 있다. Static 키워드로 지정되지 않아 공유되지 않은 필드는 인스턴스 변수로, 각자의 인스턴스마다 자신의 필드를 생성한다. Static 키워드로 생성한 정적 변수(클래스 변수)는 클래스 로더가 클래스를 메소드 영역에 적재할 때 생성된다. Static Final 로 필드를 지정하게 되면 더 이상 그 필드를 수정할 수 없다.

**객체지향 기초 문법 (정유진 정리본)**

클래스 선언의 형식은 class 클래스이름 { //필드 //메서드 }이며 ‘class’는 클래스를 선언하기 위한 키워드, ‘//필드’는 객체의 속성을 나타내고 ‘//메서드’는 객체의 동작을 나타냅니다. 보통 파일마다 1개의 클래스를 선언하지만, 2개 이상의 클래스를 1개 파일로 선언 가능합니다. 하나의 파일에 클래스가 둘 이상 있다면 하나만 public으로 선언할 수 있고, 해당 클래스 이름은 소스 파일 이름과 동일해야 합니다.

클래스의 구성 요소로는 멤버(필드 & 메서드)와 생성자로 이루어지는데 생성자는 새로운 연산자로 호출됩니다. 지역 변수는 매개변수, 즉 메서드 내부에 선언된 변수인데 예를 들어 for 문 등 블록 내부도가 있습니다. 필드는 기본 값이 있지만, 지역 변수는 기본 값이 없어 반드시 초기화를 시켜야 합니다. 필드는 클래스 전체에서 사용할 수 있지만, 지역 변수는 선언된 블록 내부 안에서 선언된 후에만 사용이 가능합니다. 필드와 메소드를 접근하기 위해서는 ‘객체참조변수.멤버’의 문법이 필요합니다. 클래스 내부에서 자신의 멤버에 접근하기 위해서는 참조 변수 this 혹은 참조 변수 없이 그냥 멤버 이름 그대로 사용하면 됩니다.

접근 제한자(제어자)에는 public, protected, default, private이 있는데 public은 모든 클래스의 접근을 허용, protected는 동일 패키지(폴더)에 존재하는 클래스의 접근을 허용, default는 동일 패키지(폴더)에 존재하는 클래스의 접근을 허용, private은 모든 클래스의 접근을 거부한다는 특징이 있습니다.

클래스 내부에 캡슐화된 멤버를 외부에서 사용할 필요가 있다면, private으로 지정된 필드는 클래스 내부에서만 사용할 수 있기 때문에 이를 위해 접근자와 설정자 사용이 필요합니다. 값을 반환하는 ‘접근자’(Getter method)는 private 필드를 외부로 반환하는 public 메소드입니다. get + 필드명의 형태로 작성하며 반환 타입은 대상 필드와 같습니다. 단, public에서 선언하면 클래스 밖의 모든 코드에서 호출 가능합니다. 값을 변경하는 ‘설정자’(Setter method)는 private 필드를 위부에서 변경하게 하는 public 메소드이며 set + 필드명의 형태로 작성하고 반환 타입은 void가 됩니다. 이 또한 public에서 선언한다면 클래스 밖의 모든 코드에서 호출 가능합니다.   
생성자의 역할은 객체를 생성하는 시점에서 필드를 다양하게 초기화한다는 점입니다. 생성자의 선언 방식은 클래스이름 ( ... ) { ... } 앞에 일반적으로 공개되어야 하므로 public으로 선언되지만 아닐 수도 있습니다. 모든 클래스는 최소한 하나의 생성자가 있어야 하고 이를 ‘디폴트 생성자’라고 합니다. 만약 생성자를 선언하지 않으면 컴파일러가 자동으로 디폴트 생성자를 추가합니다. 생성자도 메소드와 같이 오버로딩이 가능한데 이를 ‘생성자 오버로딩(Overloading)’이라 합니다.

Java는 static 키워드로 클래스의 필드를 공유할 수 있도록 지원합니다. 인스턴스 변수란 static 키워드로 지정되지 않아 공유되지 않은 필드로, 인스턴스마다 자신의 필드를 생성합니다. 정적 변수 또는 클래스 변수는 static 키워드로 지정하여 모든 인스턴스가 공유하는 필드를 말합니다. 인스턴스 변수는 객체별로 관리되는데, 객체를 생성할 때 인스턴스 변수 또한 생성되며 객체가 소멸될 때는 자동으로 소멸합니다. 정적 멤버를 활용할 때는 ‘클래스 이름.정적변수이름 ... 클래스이름.정적메서드이름()’과 같이 활용하면 되는데 상수는 변경되지 않는 변수이기 때문에 final 키워드로 지정합니다. 그러나 final로만 지정하면 객체마다 자신의 기억 공간이 형성된다는 특징이 있습니다. 또한 상수는 값이 변경되지 않으므로 객체마다 따로 기억 공간을 할당할 필요가 없습니다. 따라서 static final로 지정해서 선언해주면 됩니다.