Chap06

클래스

작성자: 강태경



# 객체 지향 프로그래밍

현실 세계는 모두 객체로 표현할 수 있고, 특징(속성) 동작(함수)를 포함한 여러 객체를 자동차 부품처럼 조립하여 코딩하는 방법

* 클래스 : 객체 구조 정의 , 객체의 설계도
* 클래스 구성 요소 : 속성(멤버변수, 필드), 생성자함수, 함수(멤버 메소드)

속성 : 객체의 고유데이터를 저장

생성자함수 : 객체의 속성을 사전 정의한 값으로 초기화함

함수 : 객체의 동작부분(기능 또는 동작)

* 객체변수 : 클래스로 부터 생성, new 생성자형태로 생성되어 저장됨, 참조자료형이므로 객체의 방번호(주소)가 저장됨
* new 연산자 : 객체를 생성하는 연산자이며, 생성자를 실행시키고, 힙 메모리 방에 객체를 넣음

# 필드(== 속성 == 멤버변수)

현실 세계는 모두 객체로 표현할 수 있고, 특징(속성) 동작(함수)를 포함한 여러 객체를 자동차 부품처럼 조립하여 코딩하는 방법

* *속성(*필드*) :* 객체의고유 *정보를 저장*
* 속성*(필드)* 선언 *: 클래스 {} 블록 어디서나 선언 가능하나, 함수 내부에서는 선언 못함*
* *속성* 사용:클래스 *내부의 생성자함수, 일반함수에서 바로 사용 가능,*
* *클래스* 외부에서 *사용할 경우, new 생성자() 통해 객체변수를 생성하고 객체변수.속성 으로 사용 가능*

# 생성자(함수)

생성자는 new 연산자로 호출되는 유사 함수임. 객체 생성 시 초기화를 담당하며 모든 클래스에 반드시 1개는 존재해야 함, 클래스에 없으면 자동 추가됨

* *기본 생성자 : 클래스 선언 시 컴파일러에 의해 자동으로 추가되는 생성자( 매개변수가 없는 생성자 )*
* *생성자 정의 : 객체를 생성할 때 호출되는 생성자를 명시적으로 선언 가능( 이때 컴파일러가 자동으로 기본 생성자를 만들어 주지 않는다 )*
* *매개 변수 : 생성자 호출 시 값을 전달받기 위해 선언되는 변수*
* *객체 속성 초기화 : 생성자 내부에서 객체의 속성을 초기화한다.*
* *오버로딩 : 매개변수 타입 및 개수를 달리하는 생성자를 여러 개 선언하는 것을 말함*
* *this : 객체 자신을 의미하는 객체변수, 클래스 내부의 생성자 및 함수에서 사용 가능*
* *this.속성 : 클래스의 생성자 및 함수에서 객체의 속성을 접근할 때 사용 가능*
* *this() : 클래스 내부에서 이미 정의된 다른 생성자를 호출할 때 사용( 중복 코딩 제거 )*

# 메소드(함수)

객체의 동작(기능)에 해당하며 실행코드가 중괄호{} 블록에 있으며, 호출되면 중괄호 블록의 모든 코드들이 일괄적으로 실행됨

* *함수 형태 :*

*접근제어자 리턴타입 함수명(매개변수타입 매개변수명1...매개변수타입 매개변수명n){실행문}*

*ex) public void sample(int a, String b, double c) {}*

* *접근제어자 : public(공용), default(생략, 패키지 내의 자바파일 허용) 등*
* *함수 이름 : 함수의 기능을 잘 표현한 단어로 정의*

*(카멜표기법(낙타표기법), 첫글자소문자 , 단어와 단어 사이는 대문자 사용)*

* *매개변수 : 함수 실행 시 전달하고 싶은 데이터*
* *리턴타입 : 함수가 실행된 이후 결과값을 {} 밖으로 전달하고 싶을 때 사용*

*return 변수; // 변수 {} 밖으로 전달됨*

* *리턴값이 없는 함수 호출 : 객체변수.함수()*
* *리턴값이 있는 함수 호출 : double result = 객체변수.함수();*

# 객체(인스턴스) 멤버와 공유(정적==전역) 멤버

객체의 동작(기능)에 해당하며 실행코드가 중괄호{} 블록에 있으며, 호출되면 중괄호 블록의 모든 코드들이 일괄적으로 실행됨

* *객체(==인스턴스)멤버 : 객체 속성(멤버변수), 객체 함수(멤버메소드)*
* *this : 객체 내부의 자기자신을 의미하는 객체변수, 자신의 클래스 안에서만 사용가능*
* *공유(==정적==전역) 변수(함수) : 객체를 생성하지 않고 사용가능, 변수(함수)앞에 static 키워드 붙임*

*사용법 : 클래스명.변수명, 클래스명.함수명()*

* *싱글톤 : 공유 객체*
* *final : 상수 만드는 키워드 , 변수의 앞에 붙이면 상수가 됨, 초기값 바로 지정해야함*

*사용법 : final 변수명 = 초기값*

* *static final : 공유 상수*

# 패키지와 접근 제한자

프로젝트 개발 시 클래스 개수가 수십 개, 수백 개로 늘어나면 관리가 어려움 발생. 체계적인 관리를 위해 패키지를 사용함(일종의 폴더와 유사)

* *패키지 선언 : 해당 클래스가 어떤 패키지에 속하는 지 선언  
  package 상위패키지.하위패키지;*
* *Import : 사용하고자 하는 클래스가 다른 패키지에 속할 경우(폴더 구조가 다른 경우) 컴파일러에게 알려주어야 함*
* *접근 제한자 : 클래스와 인터페이스를 다른 패키지에서 사용하지 못하도록 막을 필요가 있음. 객체 생성을 막거나, 속성이나 함수를 사용 못하게 막을 필요가 있을 때 사용함  
  public : 공용, 모든 객체가 사용가능  
  protected : 같은 패키지에서 자식 클래스가 아닌 클래스는 사용할 수 없음 (상속 관계)  
  default : 같은 패키지안에 있는 자바 클래스는 서로 사용 가능  
  private : 같은 패키지 뿐만 아니라 모든 외부 클래스는 사용 못함  
  (오직 자신만 사용가능)*
* *Getter/Setter : 속성은 외부에서 접근할 수 없도록 막고 함수는 공개해서 외부에서 함수를 통한 속성에 접근하도록 유도  
  Getter : 속성의 값을 외부로 리턴해주는 함수   
   명명법 : get속성명()  
   ex) getId(), getName(){}  
  Setter : 매개변수로 유효한 값만 받아 속성의 값을 변경하는 함수  
   명명법 : get속성명(매개변수타입 매개변수명){}  
   ex) setId(“forbob”), setName(“taegung”)*