38강 접근제한자 (Public)

1. 접근제한자

(1) 프로그램을 제작하다보면 클래스에 선언한 변수나 메서드를 접근하지 못하도록 막아야하는 경우有 (객체접근 제한)

→ 이러한 접근의 제한을 두기 위해 지원하는것 = 접근제한자

(2) 접근제한자는 클래스의 정보보호를 위한 캡슐화의 개념에 적용됨.

(3) 클래스 : Public, default

(4) 생성자 : Public. Protected, default, Private

(5) 멤버변수 : Public, Protected, default, Private

(6) 멤버 메서드 : Public, Protected, default, Private

(7) 지역 변수 : 접근 제한자 불가

2. 클래스의 접근제한자

(1) 저장하는 파일의 파일명을 결정하는데 사용.

(2) 하나의 자바 파일에 여러개의 클래스를 만들 경우 파일명으로 사용하는 이름은 Public이 있는 이름이며, Public은 오직 하나의 클래스에만 붙일 수 있다.

3. 생성자, 메서드, 멤버변수

(1) Public 접근제한자

⊙ Public은 아무 제약없이 자유롭게 접근이 가능한 접근 제한자이다.

(2) Protected 접근제한자

⊙ Protected는 패키지가 다르고 참조할 경우에만 접근이 불가능하다.

↓ 상속시에만 접근가능함

(3) default 접근제한자

⊙ default는 패키지가 다르면 상속이나 참조를 통해 접근이 불가능하다.

(4) Private 접근제한자

⊙ Private는 어떠한 상황이든지 모두 접근이 불가능하다.

↳ 오직 같은 class 내부에서만 사용가능

39강 캡슐화 (Encapsulation)

1. 캡슐화

(1) 클래스에 선언된 변수의 접근제어자를 private로 주고 접근을 못하게 차단한 후 접근을 가능하도록 허용하는 변수만 메서드를 통해 접근하도록 하는 방법을 캡슐화라고 부른다.

(2) 캡슐화를 이용하면 변수의 직접접근이 불가능하여 정보보호를 할 수 있으며 메서드를 통해 접근해야하기 때문에 접근에 제한을 줄 수 있다.

(3) 접근을 위한 메서드

(1) Setter → 멤버변수에 값을 넣기위해 지정하는 메서드.

(2) getter → 멤버변수에 값을 주기위해 지정하는 메서드

ex) * 클래스에서.

```
public class Class1 {
    int a = 100;
    private int b = 200;
    //Setter
    public void setB(int b){
        this.b = b;
    }
    // Getter
    public int getB() {
        return this.b;
    }
}
```

* main에서

```
c.setB(300);
System.out.println(c.getB());
```

* Source → Generate Getters and setters

40강 Static

1. Static

(1) 변수나 메서드에 Static을 붙혀주게 되면 객체의 생성 없이 사용할수있다.

(2) 같은 클래스로 부터 생성된 모든 객체들은 Static 변수를 개별적으로 가질 수 없으며, 하나의 변수를 공유로 사용한다.

(3) 변수나 매서드를 사용할 때는 클래스 이름. 멤버의 형태로 접근하여 사용한다.

(4) 왜 Static 매서드에서 사용할수 있는 멤버 변수는 Static 변수 뿐이며 Static 변수가 아닌 일반 변수들은 객체를 생성해야만 사용이 가능하기 때문

41강 Final

1. Final

⊙ 더 이상 변경할 수 없다는 의미를 가지고 있는 키워드

(1) 변수

⊙ 변수에 값을 넣을 수 없으며 변수의 선언과 동시에 초기화를 반드시 해주어야함.

(2) 매서드

⊙ 상속관계가 있을 때 자식 클래스에서 매서드를 Overriding 할 수없다.

(3) 클래스

⊙ 상속을 하지 못하도록 막아줄 수 있다.

4장 중첩클래스 (Inner Class)

Date Page

1. 중첩클래스

⊙ 클래스 내부에 클래스를 만들어서 사용하는 것.

(1) 프로그래밍을 할 때 여러군데서 사용하는 클래스가 아니라면 파일을 새로 만들거나 코드의 아래부분으로 내려서 만들지 않고 바로 만들어서 바로 쓸수 있도록 하는개념.

(2) 경우에 따라 객체를 생성할수 있는 부품이 클래스를 설계한 부분으로 한정되어 있는 경우有

(4) Inner Class ⊙ 클래스 내부에 만든 클래스
Outer class ⊙ Inner class를 감싸는 클래스

(5) 일반 중첩 클래스

① 클래스 내부에서 클래스를 만들때 사용하는 용어

② Inner class의 객체를 만들기위해서는 반드시 Outer 클래스의 객체를 생성하고 이를 통해 Inner 클래스를 만들어야한다.

③ Inner class ⟶ outer class ○ (객체가 생성되어있다는 가정하에)
Outer class ⟶ Innerclass ✕

(6) Static 중첩클래스

① 일반중첩클래스에서 Inner Class가 Static으로 정의된 경우

② Inner 클래스가 Static으로 정의되어 있어 Outer 클래스의 객체 생성없이 바로 객체생성이 가능.

③ Innerclass 에서는 Outer 클래스에 정의된 멤버 중에 static 멤버만 접근이 가능하다. → Outer 클래스의 객체가 생성되지 않았다는 가정하에 개발을 해야한다.

(7) 메서드 내부의 중첩클래스

① 클래스를 메서드 내부에 만든 경우을 의미

② 메서드 내부에 만든 중첩클래스는 다른 곳에서 참조변수조차 선언할수없다

③ Inner 클래스에서는 클래스 내부에 선언한 변수만 사용할수 있으며 클래스를 만든 메서드 내부에 있는 지역변수 중 final이 변수만 접근가능. 그 이외는 접근X.

(8) 익명 중첩 클래스

⊙ 클래스로부터 객체 생성할 때 클래스 내부 코드를 바로 작성하는것

① 부분 class가 가지고 있는 메서드를 Overriding 해야할 경우 상위받은 클래스를 만들지 않고 바로 Overriding 할수있음.

② 추상 클래스 이용시 자주 사용됨.

4장 추상클래스

1. 추상메서드

(1) 추상메서드

⊙ 클래스를 작성할 때 메서드를 구현하지 않고 선언만 해 놓은 메서드

① 구현되지 않은 메서드이므로 구현을해야 사용이 가능.

② 추상메서드는 접근제한자와 리턴타입 중간에 abstract 라는 키워드를 붙여준다.

(2) 구현된 메서드

```
public void method1(){
}
```

(3) 구현되지 않은 메서드

```
public abstract void method();
```

2. 추상클래스

⊙ 추상메서드를 하나라도 가진 클래스

(1) 추상클래스는 구현되지 않은 메서드인 추상메서드를 가지고 있어 객체의 생성이 불가능.

(2) 추상클래스의 사용을 위해서는 반드시 추상클래스를 상속받는 서브클래스 필요.

(3) 추상클래스를 상속 받은 서브 클래스는 추상 메서드이므로 반드시 구현해야 함

(4) 추상메서드 사용시 특정 메서드의 구현에 대한 강제성을 줄수있음.

```
public abstract ClassName{
}
```

44강 인터페이스

Date Page

1. 인터페이스

⊙ 추상메서드로만 구현되어 있는 것

(1) 자바에서는 단일상속만 허용되므로 추상클래스를 하나 이상 상속받을 수 ×

(2) 자바에서는 인터페이스를 통해 반드시 구현해야할 메서드를 클래스에 여러개 구현할 수있다.

(3) 인터페이스는 다중 상속을 가능하기 위해서가 아니라, 하나의 클래스를 여러종류의 참조변수를 통해 접근할 수 있고, 메서드를 호출할 수 있도록 하기위한 개념임.

(4) 인터페이스에 정의된 메서드는 모두 추상 메서드 이며, 변수는 static final 변수 이다. → 변경 ×

(5) 메서드를 선언할 때 abstract 제어자를 붙이지 않아도 추상메서드로 간주한다.

(6) 변수에 static final을 붙이지 않아도 static final로 간주한다.

(7) 인터페이스의 작성

```
Interface 인터페이스이름 {
        변수선언
        메서드선언
}
```

(8) 인터페이스의 구현

```
class 클래스이름 extends 부모클래스 Implements 인터페이스1,
인터페이스2 {        }
```