접근제어자와캡슐화

1. 접근 제어자란?

접근 제어자는 클래스, 변수, 메소드 등에 대한 접근 범위를 제어하는 키워드입니다. 이를 통해 외부에서의 무분별한 접근을 방지하고, 보안을 강화할 수 있습니다. 자바에는 네 가지 접근 제어자가 있습니다.

1) `private`

- **특징**: 해당 클래스 내부에서만 접근이 가능합니다. 외부 클래스나 다른 패키지에서는 접근할 수 없습니다.
- **사용 예**: 클래스 내부에서만 사용되는 변수나 메소드는 `private`로 선언하여 외부에서 접근하지 못하도록 합니다.

2) `public`

- **특징**: 어디서든 접근이 가능합니다. 다른 클래스, 패키지에서도 접근할 수 있습니다.
- **사용 예**: 외부에서 자주 호출될 필요가 있는 메소드나 변수는 `public`으로 선언합니다.

3) `protected`

- **특징**: 같은 패키지 내에서는 자유롭게 접근할 수 있으며, 상속받은 클래스에서도 접근이 가능합니다.
- **사용 예**: 상속받은 클래스에서 사용해야 할 메소드는 `protected`로 선언합니다.

4) 기본 접근 제어자 (`default`)

- **특징**: 같은 패키지 내에서만 접근이 가능합니다. 별도의 접근 제어자를 명시하지 않으면 기본값으로 설정됩니다.
- **사용 예**: 패키지 내에서만 사용되는 클래스나 메소드는 `default`로 선언합니다.

2. 캡슐화 (Encapsulation)

캡슐화는 클래스 내부의 데이터(변수)를 숨기고, 이를 조작하는 메소드를 통해 외부와 소통하는 방식입니다. 이를 통해 데이터를 보호하고, 객체의 무결성을 유지할 수 있습니다.

1) 예시 코드

```
public class Person {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

위 예제에서 `name` 변수는 `private`으로 선언되어 외부에서 직접 접근할 수 없습니다. 대신, `getName()` 메소드를 통해 값을 가져오고, `setName()` 메소드를 통해 값을 변경할 수 있습니다. 이와 같은 방식으로 데이터 보호가 가능하며, 이를 캡슐화라고 합니다.

2) 사용 이유

캡슐화를 사용하면 데이터를 외부에서 직접 조작하지 못하게 막을 수 있어, 객체의 상태를 안전하게 유지할 수 있습니다. 또한, 필요에 따라 메소드를 통해 데이터 검증 로직을 추가할 수 있습니다.

3. 정리

- **접근 제어자**: `private`, `public`, `protected`, 기본 접근 제어자 (default) 네 가지가 있습니다.
- **캡슐화**: 객체의 데이터를 `private`으로 숨기고, 이를 조작하는 메소드를 `public`으로 제공하여 객체의 무결성을 유지하는 기법입니다.

캡슐화와 접근 제어자는 객체 지향 프로그래밍의 중요한 개념이므로, 꼭 숙지하여 자바 코드 작성 시 올바르게 사용하시기 바랍니다.