Declaring Advice(어드바이스 선언)

"어드바이스는 특정 메서드에 대해 추가 작업을 수행하는 코드입니다. 포인트컷이라는 규칙을 사용해 어드바이스가 어떤 메서드에 적용될지를 결정합니다. 이렇게 매칭된 메서드가 실행되기 전에, 실행된 후에, 또는 실행 중에 어드바이스가 실행됩니다.

포인트컷 표현식의 방법

1. **인라인 표현식(Inline expression)** : 인라인 표현식은 프로그래밍에서 특정 규칙이나 로직을 코드 내에서 직접 정의하는 방식을 의미합니다. spectJ에서 포인트컷을 별도의 메서드나 변수로 정의하지 않고, 어드바이스(Advice) 안에서 직접적으로 작성하는 방식입니다.
   특징 :
   - 직접성: 포인트컷 인라인 표현식은 어드바이스(Advice) 내에서 직접 작성됩니다. 별도의 메서드나 변수로 분리하지 않고, 해당 어드바이스에서 바로 사용할 포인트컷 조건을 정의합니다. 예를 들어, @Before("execution(* com.example.service..(..))")와 같은 형태입니다.
   - 간결성: 인라인 표현식은 짧고 간결한 코드를 작성할 수 있게 합니다. 어드바이스와 포인트컷을 한 곳에서 정의하므로, 특정 조언이 적용될 메서드가 무엇인지 바로 알 수 있어 코드를 이해하기 쉽습니다.
   - 재사용성 부족: 인라인 표현식은 정의된 곳에서만 사용되기 때문에, 동일한 포인트컷 조건을 여러 어드바이스에서 사용하려면 매번 반복 작성해야 합니다. 이로 인해 코드 중복이 발생할 수 있으며, 재사용성이 떨어집니다.
   - 유지보수 어려움: 여러 곳에서 동일한 인라인 표현식을 사용하고 있을 경우, 조건을 수정해야 할 때 모든 인라인 표현식을 찾아 수정해야 합니다. 이로 인해 코드가 분산되어 있을 경우 유지보수가 어려워질 수 있습니다.
   - 명확성: 인라인 표현식은 어드바이스와 포인트컷이 같은 곳에 정의되기 때문에, 해당 어드바이스가 어떤 포인트컷에 적용되는지를 한눈에 파악할 수 있습니다. 이는 코드의 명확성을 높여주고, 개발자가 의도한 바를 쉽게 이해하게 합니다.
   - 사용 시기: 인라인 표현식은 간단한 포인트컷 조건을 정의하거나, 특정 조건이 반복 사용되지 않는 경우에 적합합니다. 그러나 복잡한 조건이 필요하거나 여러 어드바이스에서 동일한 포인트컷을 사용할 경우, Named Pointcut으로 정의하는 것이 더 바람직합니다. 예제코드 :

```
@Aspect
public class MyAspect {

    @Before("execution(* com.example.service.*.*(..))")
    public void beforeMethod(JoinPoint joinPoint) {
        System.out.println("Before method: " +
joinPoint.getSignature().getName());
    }
}
```

2. **named pointcut** : 재사용 가능하고 명확하게 정의된 포인트컷입니다. 포인트컷은 어떤 지점에서 어드바이스(Advice)를 적용할지를 결정하는 데 사용되며, Named Pointcut은 이러한 포인트컷을 별도의 이름을 부여하여 정의한 것입니다.