After Advice

Annotations:

- @After: Runs after the method finishes (regardless of outcome).
- @AfterThrowing: Runs after a method throws an exception.
- @AfterReturning: Runs after a method returns successfully.

Examples:

```
@Aspect
@Component
public class LoggingAspect {

   public static final Logger LOGGER=LoggerFactory.getLogger(LoggingAspect.class);

   @Before("execution (* com.telusko.springbootrest.service.JobService.getJob(..)) ||
               execution(* com.telusko.springbootrest.service.JobService.updateJob(..))")
    public void logMethodCall(JoinPoint jp) {
        LOGGER.info("Method Called "+jp.getSignature().getName());
    }


    @After("execution (* com.telusko.springbootrest.service.JobService.getJob(..)) ||
              execution(* com.telusko.springbootrest.service.JobService.updateJob(..))")
     public void logMethodExecuted(JoinPoint jp) {
         LOGGER.info("Method Executed "+jp.getSignature().getName());
     }


    @AfterThrowing("execution (* com.telusko.springbootrest.service.JobService.getJob(..)) ||
                        execution(* com.telusko.springbootrest.service.JobService.updateJob(..))")
     public void logMethodCrashed(JoinPoint jp) {
        LOGGER.info("Method has some issues "+jp.getSignature().getName());
     }


     @AfterReturning("execution (* com.telusko.springbootrest.service.JobService.getJob(..)) ||
                  execution(* com.telusko.springbootrest.service.JobService.updateJob(..))")
     public void logMethodExecutedSuccess(JoinPoint jp) {
        LOGGER.info("Method Executed Successfully "+jp.getSignature().getName());
     }
}
```