1.Maven Dependencies (pom.xml)

```
<dependencies>
   <!-- Spring Core and AOP -->
   <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.33</version>
   </dependency>
   <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-aop</artifactId>
      <version>5.3.33</version>
   </dependency>

   <!-- AspectJ -->
   <dependency>
      <groupId>org.aspectj</groupId>
      <artifactId>aspectjweaver</artifactId>
      <version>1.9.20.1</version>
   </dependency>
</dependencies>
```

---

2. OrderService.java

```
package com.shopping.service;
import org.springframework.stereotype.Component;
@Component
```

```
public class OrderService {

    public void addToCart(String product) {
        System.out.println("Adding product to cart: " + product);
    }

    public void placeOrder(String orderId) {
        if ("INVALID_ID".equals(orderId)) {
            throw new RuntimeException("OrderNotFoundException");
        }
        System.out.println("Order placed successfully for: " + orderId);
    }

    public void cancelOrder(String orderId) {
        System.out.println("Order cancelled: " + orderId);
    }
}
```

---

3. OrderLoggingAspect.java

```
package com.shopping.aspect;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.*;

import org.springframework.stereotype.Component;

@Aspect

@Component

public class OrderLoggingAspect {
```

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

    @AfterReturning(pointcut = "execution(* com.shopping.service.OrderService.*(..))",
returning = "result")
    public void logAfterReturning(JoinPoint joinPoint, Object result) {
        System.out.println("Method " + joinPoint.getSignature().getName() + " completed
successfully.");
    }

    @AfterThrowing(pointcut = "execution(* com.shopping.service.OrderService.*(..))", throwing
= "ex")
    public void logAfterThrowing(JoinPoint joinPoint, Throwable ex) {
        System.out.println(" Exception in method " + joinPoint.getSignature().getName() + ": " +
ex.getMessage());
    }

    @After("execution(* com.shopping.service.OrderService.*(..))")
    public void logAfter(JoinPoint joinPoint) {
        System.out.println("Method " + joinPoint.getSignature().getName() + " execution
finished.");
    }
}
```

---

4. Spring Config: AppConfig.java

```
package com.shopping.config;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.EnableAspectJAutoProxy;

@Configuration

@ComponentScan(basePackages = "com.shopping")

@EnableAspectJAutoProxy

public class AppConfig {

}
```

---

5. Main App for Testing

```
package com.shopping;

import com.shopping.config.AppConfig;

import com.shopping.service.OrderService;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;


public class ShoppingApp {

   public static void main(String[] args) {

      AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);

      OrderService orderService = context.getBean(OrderService.class);


      System.out.println("\n--- Valid Order ---");

      orderService.placeOrder("ORD123");
```

```
        System.out.println("\n--- Invalid Order ---");
        try {
            orderService.placeOrder("INVALID_ID");
        } catch (Exception e) {
            System.out.println("Caught exception: " + e.getMessage());
        }

        System.out.println("\n--- Add to Cart ---");
        orderService.addToCart("Laptop");

        System.out.println("\n--- Cancel Order ---");
        orderService.cancelOrder("ORD123");

        context.close();
    }
}
```

Sample Output

```
--- Valid Order ---
 Starting method: placeOrder
Order placed successfully for
```