PRACTICAL 8(E) – SPRING AOP (AFTER THROWING ADVICE)

AIM

To demonstrate Spring AOP – After Throwing Advice.

THEORY

After Throwing Advice executes when a target method throws an exception. It is mainly used for centralized exception handling, logging errors, and auditing.

SOURCE CODE – Target Class (BusinessLogic.java)

```
package aopdemo;

public class BusinessLogic {
    public void process() {
        System.out.println("Business logic started");
        throw new RuntimeException("Something went wrong");
    }
}
```

SOURCE CODE – Aspect Class (ExceptionAspect.java)

```
package aopdemo;

import org.aspectj.lang.annotation.AfterThrowing;
import org.aspectj.lang.annotation.Aspect;

@Aspect
public class ExceptionAspect {

    @AfterThrowing(
        pointcut = "execution(* aopdemo.BusinessLogic.process(..))",
        throwing = "ex")
    public void afterThrowingAdvice(Exception ex) {
        System.out.println("After Throwing advice executed");
        System.out.println("Exception message: " + ex.getMessage());
    }
}
```

SOURCE CODE – applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/aop
       http://www.springframework.org/schema/aop/spring-aop.xsd">

    <aop:aspectj-autoproxy/>
```

```
    <bean id="business" class="aopdemo.BusinessLogic"/>
    <bean class="aopdemo.ExceptionAspect"/>
</beans>
```

SOURCE CODE – MainApp.java

```
package aopdemo;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context =
            new ClassPathXmlApplicationContext("applicationContext.xml");

        try {
            BusinessLogic obj = (BusinessLogic) context.getBean("business");
            obj.process();
        } catch (Exception e) {
            System.out.println("Exception handled in main");
        }
    }
}
```

NETBEANS 7 – EXECUTION STEPS

1. Use the same Spring AOP project and libraries from Practical 8(A)
2. Modify BusinessLogic.java to throw exception
3. Create ExceptionAspect.java
4. Update applicationContext.xml
5. Run MainApp.java using Shift + F6

OUTPUT

Business logic started After Throwing advice executed Exception message: Something went wrong Exception handled in main

RESULT

Thus, Spring AOP After Throwing Advice was successfully implemented.