|  |  |  |  |
| --- | --- | --- | --- |
| **Name of Student: Pushkar Sane** | | | |
| **Roll Number: 45** | | **Lab Assignment Number: 6** | |
| **Title of Lab Assignment: Develop applications using Aspect Oriented Programming with Spring.** | | | |
| **DOP: 20-10-2023** | | **DOS: 21-10-2023** | |
| **CO Mapped:**  **CO4** | **PO Mapped:**  **PO1, PO2, PO3, PO4, PO7, PO11, PSO1, PSO2** | | **Signature:** |

**Practical No. 6**

**Aim:**

1. Before and After Advice:Create a class deposit, withdraw, beneficiary, create an aspect verification, create method payment in deposit, drawout in withdraw and add beneficiary in beneficiary class the verification aspect has method account login() and account logout(). When you run the application the account login() method should execute before payment(), drawout() and add beneficiary () a account logout() method should run after payment(), drawout() and add beneficiary().
2. Use of point cuts: Create a class deposit, withdraw, beneficiary, create an aspect verification, create method payment in deposit, drawout in withdraw and add beneficiary in beneficiary class the verification aspect has method account login() and account logout(). When you run the application the account login() method should execute before payment(), drawout() and add beneficiary () a account logout() method should run after payment(), drawout() and add beneficiary () using point cuts.
3. Create a business class multiplier(int a, int b) which returns a product of two numbers and create an aspect AdderAfterReturnAspect using After-returning advice.
4. Create a voter class with attribute name and age create an exception check if age is less than 18 throw an exception. Create an aspect of illegalVoter with a method if the voter class throws an exception the illegalVoter aspect method will run .Use Afterthrowing Advice.
5. Create a class deposit, withdraw, beneficiary, create an aspect verification, create method payment in deposit, drawout in withdraw and add beneficiary in beneficiary class the verification aspect has method account login() and account logout(). When you run the application the account login() method should execute before payment(), drawout() and add beneficiary () a account logout() method should run after payment(), drawout() and add beneficiary().

**Code:**

**Deposit.java**

package Practical 6;

public class Main {

public void payment() {

System.out.println("Payment method executed.");

}

}

**Withdraw.java**

package Practical 6;

public class Withdraw {

public void drawout() {

System.out.println("Drawout method executed.");

}

}

**Beneficiary.java**

package Practical 6;

public class Beneficiary {

public void addBeneficiary() {

System.out.println("Beneficiary added in Beneficiary class");

}

}

**VerificationAspect.java**

package Practical6;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.aspectj.lang.annotation.After;

@Aspect

public class VerificationAspect {

@Before("execution(\* aopexample.Deposit.payment(..)) || execution(\* aopexample.Withdraw.drawout(..)) || execution(\* aopexample.Beneficiary.addBeneficiary(..))")

public void accountLogin() {

System.out.println("Account login");

}

@After("execution(\* aopexample.Deposit.payment(..)) || execution(\* aopexample.Withdraw.drawout(..)) || execution(\* aopexample.Beneficiary.addBeneficiary(..))")

public void accountLogout() {

System.out.println("Account logout");

}

}

**Main.java**

package Practical6;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

Deposit deposit = (Deposit) context.getBean("deposit");

Withdraw withdraw = (Withdraw) context.getBean("withdraw");

Beneficiary beneficiary = (Beneficiary) context.getBean("beneficiary");

deposit.payment();

withdraw.drawout();

beneficiary.addBeneficiary();

}

}

**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">

<!-- Define your classes -->

<bean id="deposit" class="aopexample.Deposit" />

<bean id="withdraw" class="aopexample.Withdraw" />

<bean id="beneficiary" class="aopexample.Beneficiary" />

<!-- Define the Aspect -->

<bean id="verificationAspect" class="aopexample.VerificationAspect" />

<aop:aspectj-autoproxy />

</beans>

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>Hello\_world</groupId>

<artifactId>SpringWorld</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<!-- Important dependency -->

<version>4.11</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.14</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.14</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

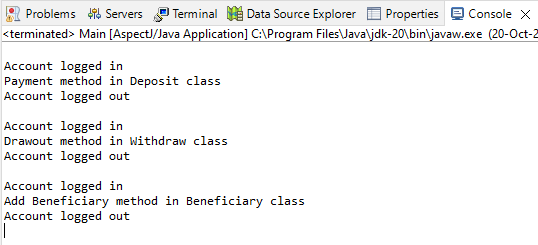
<version>1.9.7</version>

</dependency>

</dependencies>

</project>

**Output:**



1. Use of point cuts: Create a class deposit, withdraw, beneficiary, create an aspect verification, create method payment in deposit, drawout in withdraw and add beneficiary in beneficiary class the verification aspect has method account login() and account logout(). When you run the application the account login() method should execute before payment(), drawout() and add beneficiary () a account logout() method should run after payment(), drawout() and add beneficiary () using point cuts.

**Code:**

**Deposit.java**

package prac6b;

public class Deposit {

public void payment() {

System.out.println("Payment method executed.");

}

}

**Withdraw.java**

package prac6b;

public class Withdraw {

public void drawout() {

System.out.println("Drawout method executed.");

}

}

**Beneficiary.java**

package prac6b;

public class Beneficiary {

public void addBeneficiary() {

System.out.println("Beneficiary added in Beneficiary class");

}

}

**Verification.aj**

package prac6b;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.aspectj.lang.annotation.After;

import org.aspectj.lang.annotation.Pointcut;

@Aspect

public aspect Verification {

@Pointcut("execution(\* aopexample.Deposit.payment()) || execution(\* aopexample.Withdraw.drawout()) || execution(\* aopexample.Beneficiary.addBeneficiary())")

public void targetMethods() {}

@Before("targetMethods()")

public void accountLogin() {

System.out.println("Account login.");

}

@After("targetMethods()")

public void accountLogout() {

System.out.println("Account logout.");

}

}

**Main.java**

package prac6b;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("Beans.xml");

Deposit deposit = (Deposit) context.getBean("deposit");

Withdraw withdraw = (Withdraw) context.getBean("withdraw");

Beneficiary beneficiary = (Beneficiary) context.getBean("beneficiary");

deposit.payment();

withdraw.drawout();

beneficiary.addBeneficiary();

}

}

**Beans.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">

<!-- Define your classes -->

<bean id="deposit" class="aopexample.Deposit" />

<bean id="withdraw" class="aopexample.Withdraw" />

<bean id="beneficiary" class="aopexample.Beneficiary" />

<!-- Define the Aspect -->

<bean id="verificationAspect1" class="aopexample.VerificationAspect1" />

<aop:aspectj-autoproxy />

</beans>

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>prac6b</groupId>

<artifactId>prac6b</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId> <!-- Important dependency -->

<version>4.11</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.14</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.14</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

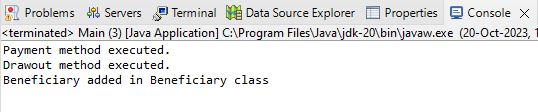
<version>1.9.7</version>

</dependency>

</dependencies>

</project>

**Output:**



1. Create a business class multiplier(int a, int b) which returns a product of two numbers and create an aspect AdderAfterReturnAspect using After-returning advice.

**Code:**

**Multiplier.java**

package prac6c;

public class Multiplier {

public int multiply(int a, int b) {

return a \* b;

}

}

**AddAfterReturnAspect.java**

package prac6c;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.AfterReturning;

@Aspect

public aspect AddAfterReturn {

@AfterReturning(pointcut = "execution(\* Multiplier.Multiplier.multiply(int, int))", returning = "result")

public void addResultToProduct(int result) {

System.out.println("After-returning advice: Result is " + result);

}

}

**MainApp.java**

package prac6c;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContect.xml");

Multiplier multiplier = (Multiplier) context.getBean("Multiplier");

int result = multiplier.multiply(5, 3);

System.out.println("Product is " + result);

}

}

**ApplicationContect.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">

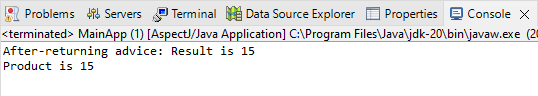
<aop:aspectj-autoproxy />

<bean id="Multiplier" class="prac6c.Multiplier" />

<bean id="addAfterReturnAspect" class="prac6c.AddAfterReturnAspect" />

</beans>

**Output:**

****

1. Create a voter class with attribute name and age create an exception check if age is less than 18 throw an exception. Create an aspect of illegalVoter with a method if the voter class throws an exception the illegalVoter aspect method will run .Use Afterthrowing Advice.

**Code:**

**Voter.java**

package aop\_voter;

public class voter {

private String name;

private int age;

public void setName(String name) {

this.name = name;

}

public void setAge(int age) {

if (age < 18) {

throw new IllegalArgumentException("Age must be at least 18 to vote");

} else {

System.out.println("You are eligible to vote");

}

this.age = age;

}

}

**IllegalVoterAspect.java**

package aop\_voter;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.AfterThrowing;

@Aspect

public class IllegalVoterAspect {

@AfterThrowing(pointcut = "execution(\* aop\_voter.voter.setAge(int))", throwing = "ex")

public void handleIllegalVoterException(Exception ex) {

System.out.println("After-throwing advice: " + ex.getMessage()); } }

**MainApp.java**

package aop\_voter;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

try {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

voter voter1 = (voter) context.getBean("voter");

voter1.setName("Pushkar");

voter1.setAge(16);

} catch (Exception e) {

// Exception is handled by the aspect

}

}

}

**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="voter" class="aop\_voter.voter" />

<bean id="illegalVoterAspect" class="aop\_voter.IllegalVoterAspect" />

</beans>

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>Voter</groupId>

<artifactId>AOP\_voter</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId> <!-- Important dependency -->

<version>4.11</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.14</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.14</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.7</version>

</dependency>

</dependencies>

</project>

**Output:**

