Міністерство освіти і науки України

Черкаський державний технологічний університет

Кафедра комп’ютерних наук та системного аналізу

Звіт з лабораторної роботи №3

з дисципліни «Розробка Web-додатків на Java з використанням фреймворку Spring»

Перевірив: Виконав:

асистент кафедри ПЗАС студент групи КН-2001

Немов Р.Г. Заєць Володимир

Черкаси 2023

Тема: Аспектно орієнтоване програмування.

Мета: Навчитися використовувати аспектно орієнтоване програмування при розробці.

Завдання: У проекті який створювали у 1-2 л.р. необхідно реалізувати:

1. Мінімум 1 Aspect клас

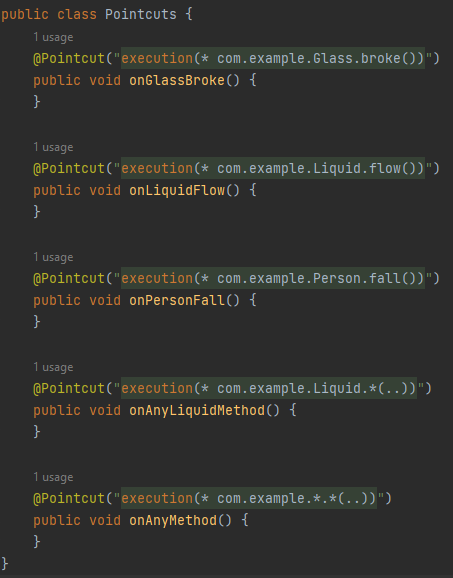
2. Мінімум 3 Advice методи

3. Мінімум 3 Pointcut

4\*(не обов'язкове). 1 комбінований Pointcut.

Хід роботи:

Для початку напишемо клас Pointcuts.java:



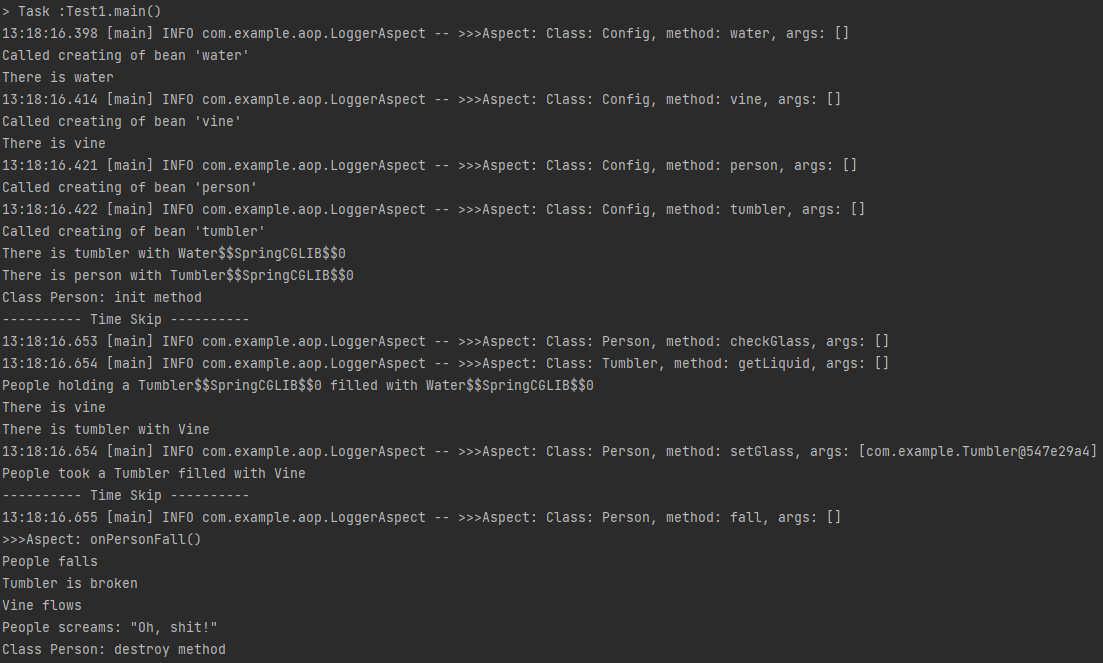
Тепер напишемо клас LoggerAspect.java з Advice для Pointcuts:

package com.example.aop;  
  
import lombok.SneakyThrows;  
import lombok.extern.slf4j.Slf4j;  
import org.aspectj.lang.ProceedingJoinPoint;  
import org.aspectj.lang.annotation.Around;  
import org.aspectj.lang.annotation.Aspect;  
import org.aspectj.lang.annotation.Before;  
import org.springframework.stereotype.Component;  
  
import java.util.Arrays;  
import java.util.List;  
  
@Component  
@Aspect  
@Slf4j  
public class LoggerAspect {  
  
 @Before("com.example.aop.Pointcuts.onGlassBroke()")  
 public void onGlassBroke() {  
 System.*out*.println(">>>Aspect: onGlassBroke()");  
 }  
  
 @Before("com.example.aop.Pointcuts.onLiquidFlow()")  
 public void onLiquidFlow() {  
 System.*out*.println(">>>Aspect: onLiquidFlow()");  
 }  
  
 @Before("com.example.aop.Pointcuts.onPersonFall()")  
 public void onPersonFall() {  
 System.*out*.println(">>>Aspect: onPersonFall()");  
 }  
  
 @Before("com.example.aop.Pointcuts.onAnyLiquidMethod()")  
 public void onAnyLiquidMethod() {  
 System.*out*.println(">>>Aspect: onAnyLiquidMethod()");  
 }  
  
 @SneakyThrows  
 @Around("com.example.aop.Pointcuts.onAnyMethod()")  
 public Object onAnyMethod(ProceedingJoinPoint joinPoint) {  
 String className = joinPoint.getSignature().getDeclaringType().getSimpleName();  
 String methodName = joinPoint.getSignature().getName();  
 List<String> args = Arrays.*stream*(joinPoint.getArgs()).map(Object::toString).toList();  
  
 LogInfo logInfo = new LogInfo(className, methodName, args);  
  
 *log*.info(">>>Aspect: Class: " + logInfo.className  
 + ", method: " + logInfo.methodName  
 + ", args: " + args);  
  
 return joinPoint.proceed();  
 }  
  
 record LogInfo(String className, String methodName, List<String> args) {  
 }  
}

Тепер напишемо тест, в якому перевіримо послідовність створення та функціональність бінів:



Результат виконаня тесту аспектів:



Посилання на код на GitHub:

<https://github.com/volodymyr-zaiets/Java-for-Web-Spring/tree/main/Lab3>

Висновок: В результаті виконання лабораторної роботи навчився використовувати аспектно орієнтоване програмування при розробці.