Министерство транспорта Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»

РУТ (МИИТ)

Институт управления и цифровых технологий

Кафедра «Цифровые технологии управления транспортными процессами»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

**ТЕМА: «СОЗДАНИЕ КОМПОНЕНТА CDI»**

**по дисциплине «Корпоративные информационные системы»**

**Вариант №8**

Выполнил студент группы УВА-411 И. Д. Лёвин

Принял преподаватель кафедры ЦТУТП П. О. Козьяков

Москва 2022

**Текст задания на лабораторную работу**

**Вариант 8.** Создать компонент CDI, выполняющий нахождение высоты треугольника различными способами (по основанию и площади, с помощью углов и сторон и т.п.). Выбор способа нахождение высоты треугольника должен быть реализован через указание соответствующей аннотации или посредством указания класса в XML файле. Посредством перехватчиков провести сравнение по времени эффективности различных методов.

**Текст программы на Java**

Класс SquareSide.java

package ru.miit.ProgrammingJ2EE.SortingMethods;  
  
import java.util.List;  
  
@SQUARE\_SIDE  
 //@Alternative  
public class SquareSide implements IHeightSearch {  
 @Override  
 public List<Double> sort(List<Double> list) {  
 System.*out*.println("Подсчет высоты по площади и стороне");  
 Double s = 0.5\* list.get(0)\*list.get(1)\*Math.*sin*( Math.*toRadians*(list.get(2)));  
 Double c = Math.*pow*(Math.*pow*(list.get(0),2)+Math.*pow*(list.get(1),2)-2\*list.get(0)\*list.get(1)\*Math.*cos*(Math.*toRadians*(list.get(2))),0.5);  
 list.set(0,2\*s/list.get(0));  
 list.set(1,2\*s/list.get(1));  
 list.set(2,2\*s/c);  
 return list;  
 }  
}

Класс ThreeLong.java

package ru.miit.ProgrammingJ2EE.SortingMethods;  
  
import javax.enterprise.inject.Default;  
import java.util.List;  
  
  
@Default  
public class ThreeLong implements IHeightSearch {  
 @Override  
 public List<Double> sort(List<Double> list) {  
 System.*out*.println("Подсчет высоты через длины всех сторон");  
 Double c = Math.*pow*(Math.*pow*(list.get(0),2)+Math.*pow*(list.get(1),2)-2\*list.get(0)\*list.get(1)\*Math.*cos*(Math.*toRadians*(list.get(2))),0.5);  
 Double p = (list.get(0)+list.get(1)+c)/2;  
 list.set(0,2\*Math.*pow*(p\*(p-list.get(0))\*(p-list.get(1))\*(p-c),0.5)/list.get(0));  
 list.set(1,2\*Math.*pow*(p\*(p-list.get(0))\*(p-list.get(1))\*(p-c),0.5)/list.get(1));  
 list.set(2,2\*Math.*pow*(p\*(p-list.get(0))\*(p-list.get(1))\*(p-c),0.5)/c);  
 return list;  
 }  
}

Класс SideAngle.java

package ru.miit.ProgrammingJ2EE.SortingMethods;  
  
import javax.enterprise.inject.Alternative;  
import java.util.List;  
  
@Alternative  
public class SideAngle implements IHeightSearch {  
 @Override  
 public List<Double> sort(List<Double> list){  
 System.*out*.println("Подсчет высоты через длину прилежащей стороны и синус угла");  
 Double c = Math.*pow*(Math.*pow*(list.get(0),2)+Math.*pow*(list.get(1),2)-2\*list.get(0)\*list.get(1)\*Math.*cos*(Math.*toRadians*(list.get(2))),0.5);  
 Double sina=(Math.*pow*(list.get(0),2)+Math.*pow*(c,2)-Math.*pow*(list.get(1),2))/(2\*list.get(0)\*c);  
 list.set(0,list.get(0)\*Math.*sin*(Math.*toRadians*(list.get(2))));  
 list.set(1,list.get(1)\*Math.*sin*(Math.*toRadians*(180-sina-list.get(2))));  
 list.set(2,c\*Math.*sin*(Math.*toRadians*(sina)));  
 return list;  
 }  
}

Интерфейс IHeightSearch.java

package ru.miit.ProgrammingJ2EE.SortingMethods;  
  
import java.util.List;  
  
  
public interface IHeightSearch {  
 List<Double> sort(List<Double> list);  
}

Аннотация SQUARE\_SIDE

package ru.miit.ProgrammingJ2EE.SortingMethods;  
  
import javax.inject.Qualifier;  
import java.lang.annotation.ElementType;  
import java.lang.annotation.Retention;  
import java.lang.annotation.RetentionPolicy;  
import java.lang.annotation.Target;  
  
@Qualifier  
@Retention(RetentionPolicy.*RUNTIME*)  
@Target({ElementType.*FIELD*, ElementType.*TYPE*, ElementType.*METHOD*})  
public @interface SQUARE\_SIDE {  
}

Класс-перехватчик ProfileInterceptor.java

package ru.miit.ProgrammingJ2EE.Interceptors;  
  
import javax.interceptor.AroundInvoke;  
import javax.interceptor.InvocationContext;  
  
public class ProfileInterceptor {  
 @AroundInvoke  
 public Object profile(InvocationContext ic) throws Exception {  
 long initTime = System.*nanoTime*();  
 try {  
 return ic.proceed();  
 } finally {  
 Double diffTime = ((System.*nanoTime*() - initTime)/1000000.0);  
 System.*out*.println(ic.getMethod() + " took " + diffTime + " millis");  
 }  
 }  
  
}

Класс-компонент IHeightSearch.java

package ru.miit.ProgrammingJ2EE.Component;  
  
import ru.miit.ProgrammingJ2EE.Interceptors.ProfileInterceptor;  
import ru.miit.ProgrammingJ2EE.SortingMethods.IHeightSearch;  
import ru.miit.ProgrammingJ2EE.SortingMethods.SQUARE\_SIDE;  
  
import javax.inject.Inject;  
import javax.interceptor.Interceptors;  
import java.util.List;  
  
@Interceptors(ProfileInterceptor.class) // Перехватчик  
public class CollectionSort<T> {  
 */\*\*  
 \* Внедрение аннотации. Точка внедрения.  
 \*/* @Inject  
 @SQUARE\_SIDE  
 private IHeightSearch collectionSorter;  
  
 public List<Double> sort(List<Double> list) {  
 collectionSorter.sort(list);  
 return list;  
 }  
}

Lab1Start.java

package ru.miit.ProgrammingJ2EE;  
  
import org.jboss.weld.environment.se.Weld;  
import org.jboss.weld.environment.se.WeldContainer;  
import ru.miit.ProgrammingJ2EE.Component.CollectionSort;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
public class lab1Start {  
 protected static Weld *weld*;  
 protected static WeldContainer *container*;  
  
 public static void main(String args[]) throws Exception {  
 *weld* = new Weld();  
 *container* = *weld*.initialize();  
  
 try {  
 CollectionSort collectionSort = *container*.instance().select(CollectionSort.class).get();  
  
 List<Double> integerList = new ArrayList<Double>();  
 integerList.add(new Double(Math.*random*()\*9+1));  
 integerList.add(new Double(Math.*random*()\*9+1));  
 integerList.add(new Double(Math.*random*()\*178+1));  
 for(int i = 0; i < 100; i++) {  
 integerList.set(0,new Double(Math.*random*()\*9+1));  
 integerList.set(1,new Double(Math.*random*()\*9+1));  
 integerList.set(2,new Double(Math.*random*()\*178+1));  
 System.*out*.println("сторона, сторона, угол");  
 System.*out*.println(integerList);  
 collectionSort.sort(integerList);  
 System.*out*.println("высота при a, высота при b, высота при c");  
 System.*out*.println(integerList);  
 }  
 }  
 finally {  
 *weld*.shutdown();  
 }  
 }  
}

Дескриптор развёртывания beans.xml

<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/beans\_1\_1.xsd"  
 version="1.1" bean-discovery-mode="all">  
  
 <alternatives>  
 <class>ru.miit.ProgrammingJ2EE.SortingMethods.SideAngle</class>  
 </alternatives>

</beans>

**Сравнение эффективностей методов**

|  |  |  |  |
| --- | --- | --- | --- |
| Количество итераций | Время работы метода (мс) | | |
| Подсчет высоты через длины всех сторон | Подсчет высоты по площади и стороне | Подсчет высоты через длину прилежащей стороны и синус угла |
| 10 | 0.02029 | 0.03783 | 0.1696 |
| 50 | 0.027558 | 0.028027 | 0.0877 |
| 200 | 0.030457 | 0.026379 | 0.0689 |

**Заключение**

В ходе лабораторной работы был изучен механизм создания CDI компонента, проанализирован и протестирован на практике механизм работы перехватчиков и применение альтернатив с использованием дескриптора развертывания. Для подсчета времени выполнении способов подсчета высоты треугольника используется перехватчик. На основании результатов сравнения эффективностей алгоритмов можно сделать вывод о том, лучший способ для подсчета высоты по площади и стороне.