Лабораторна робота №11

Спадкування.Інтерфейси.Колекції пакету java.util

***Мета робот:***

- набуття практичних навичок розробки проектів мовою Java на

прикладі простого проекту в середовищі Eclipse;

- використання шаблону проектування Factory Method;

**1 ЗАГАЛЬНЕ ЗАВДАННЯ**

В качестве основы использовать исходный текст проекта предыдущей

лабораторной работы. Обеспечить размещение результатов вычислений в

коллекции с возможностью сохранения/восстановления.

Используя шаблон проектирования Factory Method (Virtual Constructor),

разработать иерархию, предусматривающую расширение за счёт добавления новых отображаемых классов. Расширить иерархию интерфейсом "фабрикуемых" объектов, представляющим набор методов для отображения результатов вычислений. Реализовать эти методы для вывода результатов в текстовом виде. Разработать и

реализовать интерфейс для "фабрикующего" метода.

Обеспечить диалоговый интерфейс с пользователем.

Разработать класс для тестирования основной функциональности.

Использовать комментарии для автоматической генерации документации

средствами javadoc.

**2 ОПИС ПРОГРАМИ**

**2.1 Ієрархія та структура класів**

**class Info-організовує діалог з користувачем**

**class Calculator виконує індивідуальне завдання**

**class Serializator виконує збереження данних классу Calculator у файл**

**interface Calculator- містить основні методи калькуляторів**

**class Calculator Area –рахує площу**

**class Calculator FactoreArea – створює об єкт типу CalculatorArea**

**class Calculator Perimetr – рахує періметр індивідуального завдання**

**class Calculator FactoryPerimetr- створює об єкт типу CalculatorPerimetr**

**2.2 Опис програми**

На рисунку 2.1 наведена структура розробленої програми

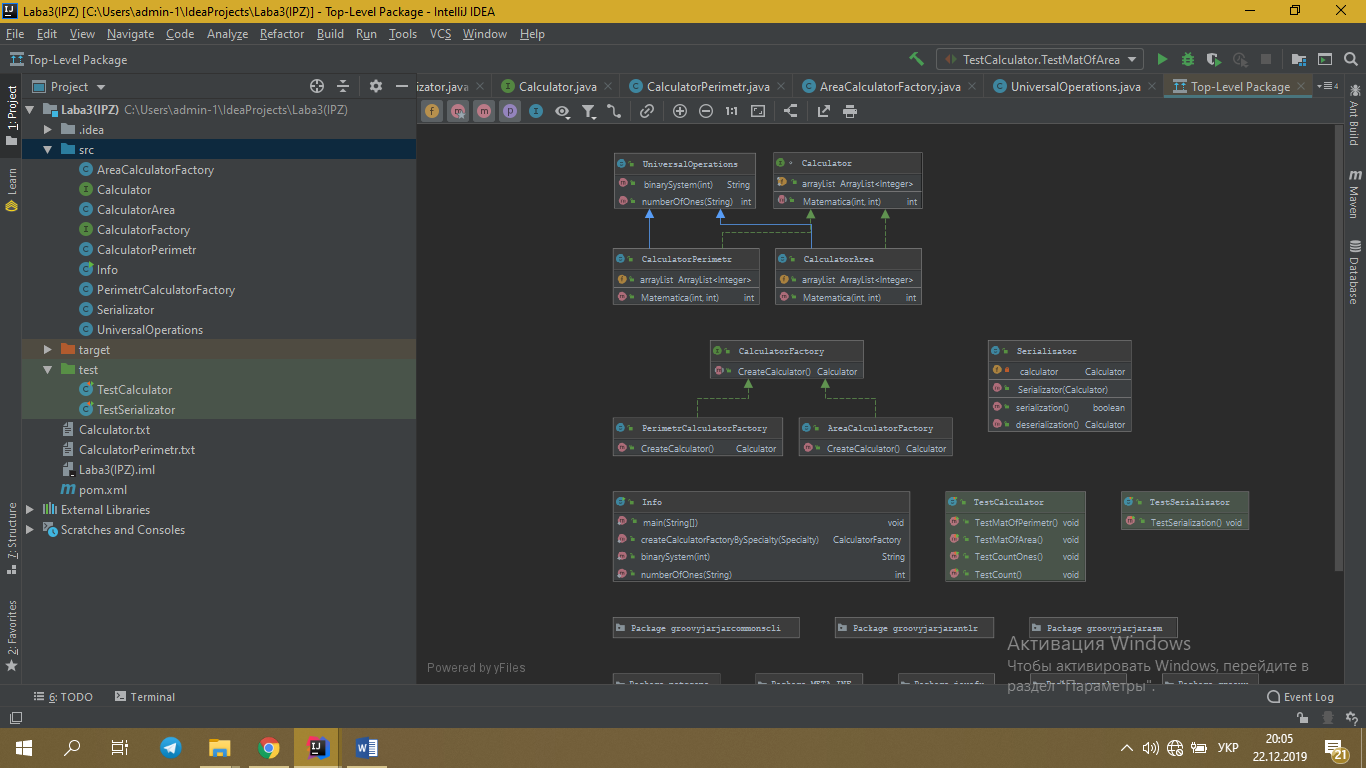


Рис 2.1 – структура програми

**2.3 Важливі фрагменти програми**

2.3.1 Файл*Calculator.java*

import java.util.ArrayList;  
  
*/\*\*  
 \* The interface Calculator.  
 \*/*interface Calculator {  
  
 */\*\*  
 \* The constant arrayList.  
 \*/* public ArrayList<Integer> *arrayList*=new ArrayList<Integer>();  
  
 */\*\*  
 \* Matematica int.  
 \*  
 \** ***@param*** *h the h  
 \** ***@param*** *side the side  
 \** ***@return*** *the int  
 \*/* public int Matematica( int h,int side);  
  
  
}

2.3.1 Файл *CalculatorFactory.java*

public interface CalculatorFactory {  
  
public Calculator CreateCalculator();  
  
  
}

2.3.1 Файл *CalculatorPerimetr.java*

import java.io.Serializable;  
import java.lang.reflect.Array;  
import java.util.ArrayList;  
  
*/\*\*  
 \* The type Calculator.  
 \*/*public class CalculatorPerimetr extends UniversalOperations implements Serializable ,Calculator{  
public ArrayList<Integer> arrayList=new ArrayList<Integer>();  
  
 */\*\*  
 \* Matematica int.  
 \*  
 \** ***@param*** *h the h  
 \** ***@param*** *side the side  
 \** ***@return*** *the int  
 \*/* public int Matematica( int h,int side)  
{  
 if (h<0||side<0)  
 {  
 h=0;  
 side=0;  
 }  
  
  
  
double rectangular=(h\*2)+(side\*2);  
double triangle=(Math.*sqrt*(Math.*pow*((side/2),2)+Math.*pow*(h,2)))\*2+side;  
 double sum= rectangular+triangle;  
  
  
 String s=Double.*toString*(sum);  
  
 s= s.substring(0,s.indexOf('.'));  
 Integer res=Integer.*parseInt*(s);  
arrayList.add(res);  
 return res ;  
}  
  
  
  
}

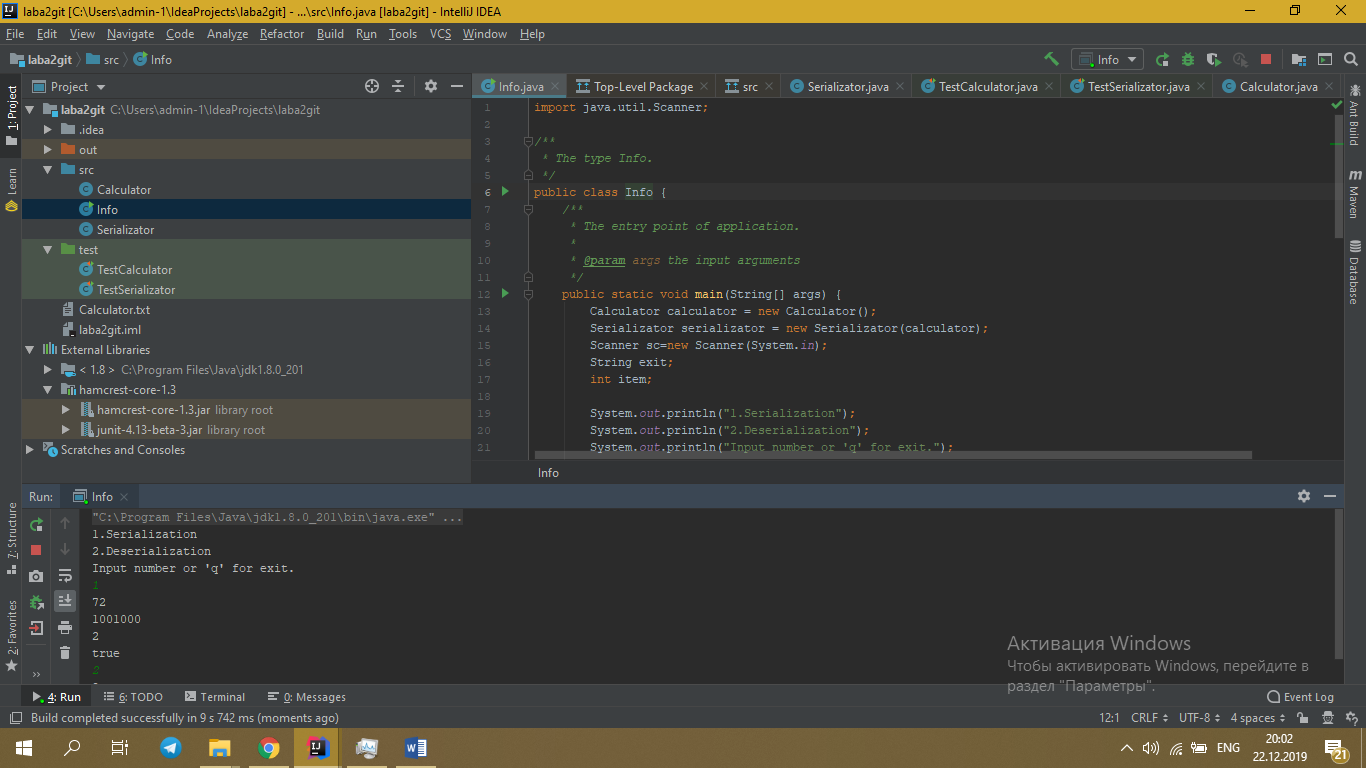
2.3.1 Файл *CalculatorTest.java*

import org.junit.Assert;  
import org.junit.Test;  
  
*/\*\*  
 \* The type Test calculator.  
 \*/*public class TestCalculator {  
 */\*\*  
 \* Test mat.  
 \*/* @Test  
 public void TestMat() {  
 int result=0;  
 Calculator calculator =new Calculator();  
  
  
 result =calculator.Matematica(8,12);  
 Assert.*assertEquals*(72,result);  
  
  
 result =calculator.Matematica(8,6);  
 Assert.*assertEquals*(51,result);  
  
  
 result =calculator.Matematica(0,0);  
 Assert.*assertEquals*(0,result);  
  
  
 result =calculator.Matematica(-1,-8);  
 Assert.*assertEquals*(0,result);  
 }  
  
 */\*\*  
 \* Test count ones.  
 \*/* @Test  
 public void TestCountOnes() {  
 String result;  
 Calculator calculator =new Calculator();  
  
  
  
 result= calculator.CountOnes(72);  
 Assert.*assertEquals*("1001000",result);  
  
  
  
 result= calculator.CountOnes(0);  
 Assert.*assertEquals*("",result);  
  
  
  
 result= calculator.CountOnes(1);  
 Assert.*assertEquals*("1",result);  
  
  
 result= calculator.CountOnes(7);  
 Assert.*assertEquals*("111",result);  
  
  
 result= calculator.CountOnes(1000);  
 Assert.*assertEquals*("1111101000",result);  
  
 result= calculator.CountOnes(2423);  
 Assert.*assertEquals*("100101110111",result);  
  
 }  
  
 */\*\*  
 \* Test count.  
 \*/* @Test  
 public void TestCount()  
 {  
 int result;  
 Calculator calculator =new Calculator();  
  
  
 result =calculator.Count("1111");  
 Assert.*assertEquals*(4,result);  
  
  
  
 result =calculator.Count("1110");  
 Assert.*assertEquals*(3,result);  
  
 result =calculator.Count("1010");  
 Assert.*assertEquals*(2,result);  
  
 result =calculator.Count("0010");  
 Assert.*assertEquals*(1,result);  
  
 result =calculator.Count("0");  
 Assert.*assertEquals*(0,result);  
  
 result =calculator.Count("1");  
 Assert.*assertEquals*(1,result);  
  
  
 result =calculator.Count("1-+ 010");  
 Assert.*assertEquals*(2,result);  
 }  
  
}

2.3.1 Файл *SerializatorTest.java*

import org.junit.Assert;  
import org.junit.Test;  
  
*/\*\*  
 \* The type Test serializator.  
 \*/*public class TestSerializator {  
  
 */\*\*  
 \* Test serialization.  
 \*/* @Test  
 public void TestSerialization()  
 {  
 boolean oper;  
 Calculator calculator=new Calculator();  
  
 Serializator serializator=new Serializator(calculator);  
 oper= serializator.serialization();  
 Assert.*assertEquals*(true,oper);  
 System.*out*.println(calculator.Count(calculator.CountOnes(calculator.Matematica(8, 12))));  
  
  
 Calculator calculator1 =new Calculator();  
 calculator1= serializator.deserialization();  
 int i;  
 i= calculator1.getCount();  
 Assert.*assertEquals*(2,i);  
 }  
  
  
}

**3 РЕЗУЛЬТАТИ**



**ВИСНОВКИ**

Навчились працювати тестами та шаблони.