Лабораторна робота №10

Класи та об'єкти. Агрегування. Серіалізація

***Мета робот:***

- набуття практичних навичок розробки проектів мовою Java на

прикладі простого проекту в середовищі Eclipse;

- демонстрація принципів агрегування при розробці класів;

- використання можливості серіалізації об'єктів.

**1 ЗАГАЛЬНЕ ЗАВДАННЯ**

Розробити клас, що серіалізується, для зберігання параметрів і результатів обчислень.Використовуючи агрегування, розробити клас для знаходження рішення задачі. Розробити клас для демонстрації в діалоговому режимі збереження та відновлення стану об'єкта, використовуючи серіалізацію. Показати особливості використання transient полів. Розробити клас для тестування коректності результатів обчислень та серіалізації/десеріалізації. Використовувати докладні коментарі для автоматичної генерації документації засобами javadoc.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

Номер індивідуального завдання визначається відповідно до номера в

журналі академічної групи за формулою:

TaskNumber = ((N – 1) % K) + 1;

де:

N – номер у журналі академічної групи;

K – кількість варіантів індивідуальних завдань;

% – операція отримання залишку від ділення**.**

Определить количество 1 в двоичном представлении целой части значения суммы периметров равнобедренного треугольника и прямоугольника по заданной высоте и длине стороны основания.

**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**

**2.2 Опис програми**

На рисунку 2.1 наведена структура розробленої програми

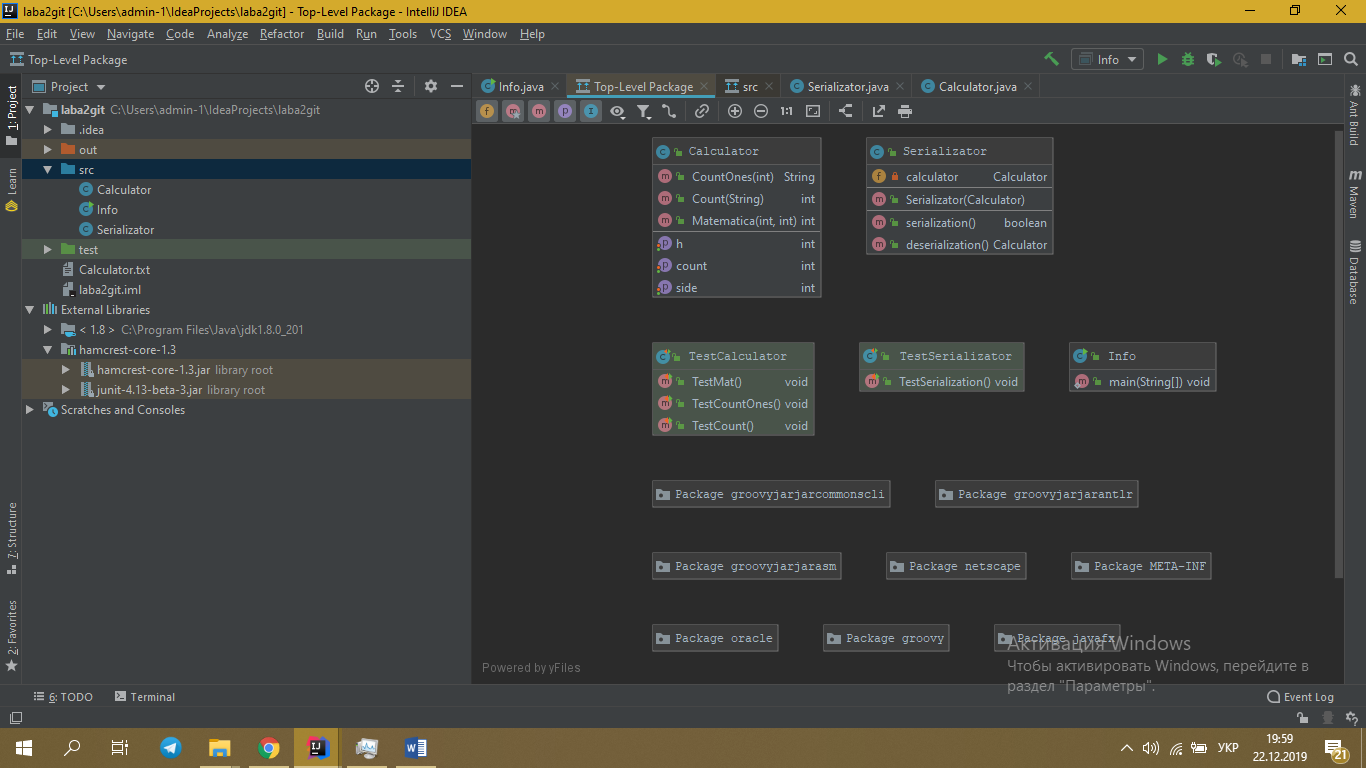


Рис 2.1 – структура програми

**2.3 Важливі фрагменти програми**

2.3.1 Файл*Calculator.java*

import java.io.Serializable;  
  
*/\*\*  
 \* The type Calculator.  
 \*/*public class Calculator implements Serializable {  
private int count;  
private transient int h=0;  
private transient int side=0;  
  
 */\*\*  
 \* Sets count.  
 \*  
 \** ***@param*** *count the count  
 \*/* public void setCount(int count) {  
 this.count = count;  
 }  
  
 */\*\*  
 \* Gets h.  
 \*  
 \** ***@return*** *the h  
 \*/* public int getH() {  
 return h;  
 }  
  
 */\*\*  
 \* Sets h.  
 \*  
 \** ***@param*** *h the h  
 \*/* public void setH(int h) {  
 this.h = h;  
 }  
  
 */\*\*  
 \* Gets side.  
 \*  
 \** ***@return*** *the side  
 \*/* public int getSide() {  
 return side;  
 }  
  
 */\*\*  
 \* Sets side.  
 \*  
 \** ***@param*** *side the side  
 \*/* public void setSide(int side) {  
 this.side = side;  
 }  
  
 */\*\*  
 \* Gets count.  
 \*  
 \** ***@return*** *the count  
 \*/* public int getCount() {  
 return count;  
 }  
  
 */\*\*  
 \* Count ones string.  
 \*  
 \** ***@param*** *i the  
 \** ***@return*** *the string  
 \*/* public String CountOnes(int i)  
{  
  
 String value="";  
 if (i==0)  
 {  
 return value;  
 }  
 while(i!=1) {  
 if (i % 2 == 0) {  
 value+="0";  
 }  
 else {  
 value+="1";  
 }  
 i/=2;  
 }  
 value+="1";  
  
  
 String reverse = new StringBuffer(value).reverse().toString();  
  
  
return reverse;  
}  
  
 */\*\*  
 \* Count int.  
 \*  
 \** ***@param*** *s the s  
 \** ***@return*** *the int  
 \*/* public int Count(String s)  
{  
 int count1=0;  
 for (int i = 0; i <s.length() ; i++) {  
 if (s.charAt(i)=='1')  
 {  
 ++count1;  
 }  
  
 }  
 count=count1;  
 return count1;  
}  
  
 */\*\*  
 \* 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);  
  
 return res ;  
}  
  
  
  
}

2.3.1 Файл *Info.java*

import java.util.Scanner;  
  
*/\*\*  
 \* The type Info.  
 \*/*public class Info {  
 */\*\*  
 \* The entry point of application.  
 \*  
 \** ***@param*** *args the input arguments  
 \*/* public static void main(String[] args) {  
 Calculator calculator = new Calculator();  
 Serializator serializator = new Serializator(calculator);  
 Scanner sc=new Scanner(System.*in*);  
 String exit;  
 int item;  
  
 System.*out*.println("1.Serialization");  
 System.*out*.println("2.Deserialization");  
 System.*out*.println("Input number or 'q' for exit.");  
 while (!(exit = sc.next()).equals("q")) {  
 try {  
 item = Integer.*parseInt*(exit);  
  
 switch (item) {  
 case 1:  
  
 System.*out*.println(calculator.Matematica(8, 12));  
 System.*out*.println(calculator.CountOnes(calculator.Matematica(8, 12)));  
 System.*out*.println(calculator.Count(calculator.CountOnes(calculator.Matematica(8, 12))));  
  
  
 System.*out*.println(serializator.serialization());  
 break;  
  
  
  
  
  
  
 case 2:  
 Calculator d = serializator.deserialization();  
 System.*out*.println(d.getCount());  
  
 break;  
 }  
 } catch (NumberFormatException e) {  
 System.*out*.println("Input number or 'q' for exit.");  
 }  
 }  
 }  
}

2.3.1 Файл *Info.java*

import java.io.\*;  
  
*/\*\*  
 \* The type Serializator.  
 \*/*public class Serializator {  
 private Calculator calculator;  
  
 */\*\*  
 \* Instantiates a new Serializator.  
 \*  
 \** ***@param*** *calculator the calculator  
 \*/* public Serializator(Calculator calculator)  
 {  
 this.calculator=calculator;  
 }  
  
 */\*\*  
 \* Serialization boolean.  
 \*  
 \** ***@return*** *the boolean  
 \*/* public boolean serialization()  
 {  
 boolean operation=false;  
 try {  
 FileOutputStream fileOutputStream=new FileOutputStream("Calculator.txt");  
 ObjectOutputStream objectOutputStream=new ObjectOutputStream(fileOutputStream);  
 objectOutputStream.writeObject(calculator);  
  
 if (objectOutputStream!=null) {  
  
 objectOutputStream.close();  
 operation=true;  
 }  
  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return operation;  
 }  
  
 */\*\*  
 \* Deserialization calculator.  
 \*  
 \** ***@return*** *the calculator  
 \*/* public Calculator deserialization()  
 {  
 FileInputStream fileInputStream = null;  
 try {  
 fileInputStream = new FileInputStream("Calculator.txt");  
 ObjectInputStream objectInputStream =new ObjectInputStream(fileInputStream);  
 Calculator calculator= (Calculator) objectInputStream.readObject();  
  
 objectInputStream.close();  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
return calculator;  
 }  
}

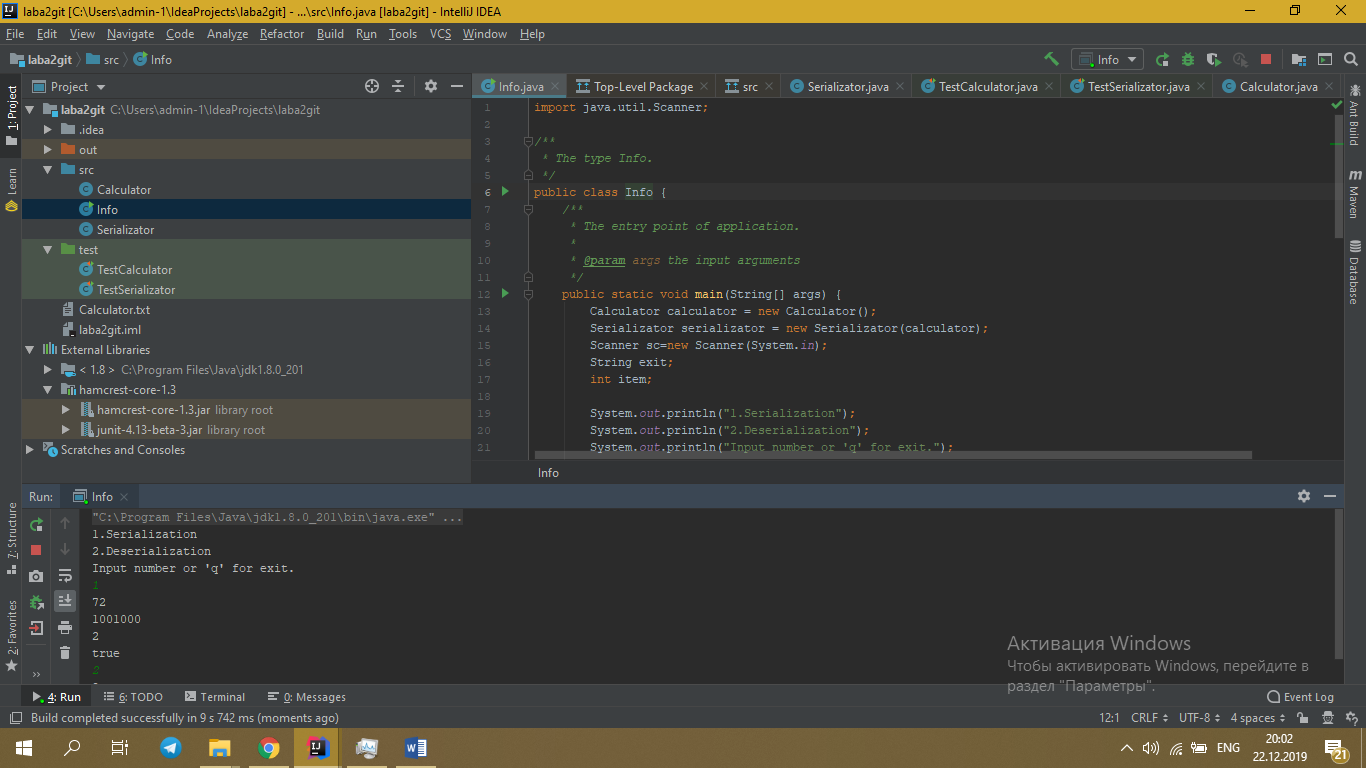
2.3.1 Файл *CalculatorTest.java*

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 використовувати засоби ООП.