软件学院 软件构造 课程实验报告

**2017 ～2018学年 第 一 学期 2017 级 软件 专业**

班级： 2017204 学号： 201720419 姓名： 周亚涛

## 实验二 面向对象软件构造

**一、实验目的**

1、掌握面向对象设计方法。

2、掌握类设计方法。

3、掌握面向对象自动化测试工具JUnit。

**二、实验环境与实验要求**

1. PC机一台，VC6.0；

2、实验前做好上机实验的准备，按照教材3.6节内容完成构造任务；

3、使用visio设计工具进行OOD，完成UML设计图绘制；

4、使用eclipse编写面向对象的程序；

4、使用JUnit完成类的单元测试；

3、实验后做好实验总结，根据实验情况完成总结报告。

**三、实验内容**

1、设计一构造-版本1.0

1）编写算式类

1.1 设计算式类的静态类图（使用visio的UML静态类图设计工具）

1.2 编写类的成员变量、成员函数

1.3 编写算式测试类

1.4 对算式类进行单元测试

1. 编写习题类

2.1 设计习题类的静态类图

2.2 编写习题类中的成员变量和成员函数

2.3 编写习题测试类

2.4 对习题类进行单元测试

1. 设计二构造-版本1.1

设计满足SOLID的面向对象程序。

1. 设计编写算是类，对算式类进行抽象，设计父类和子类的继承关系

3.1 编写算式抽象类

3.2 编写加法算式子类

3.3 编写减法算式子类

3.4 改写算式测试类，在其中加入加法测试和减法测试方法

3.5 执行单元测试

1. 设计编写习题类和习题集系统类

4.1 编写习题类

4.2 编写习题类中的产生一定数量习题的方法

4.3 编写格式化显示方法

4.4 编写一定数量的减法方法

4.5 编写一定数量的加法方法

4.6 编写习题集系统类，为程序提供Main入口。

3) 完成50道加法、50道减法和50道加减法混合习题生成和格式化显示，直至单元测试通过。

**四、实验过程**

**UML类图：**





1.

**package** cbsc.cha3;

**import** java.util.Random;

**public** **class** BinaryOperation\_3\_1 {

**static** **final** **int** ***UPPER*** = 100;

**static** **final** **int** ***LOWER*** = 0;

**private** **int** left\_operand=0, right\_operand=0;

**private** **char** operator='+';

**private** **int** value=0;

// 不是构造器

**private** **void** construct (**int** left, **int** right, **char** op) {

left\_operand = left;

right\_operand = right;

operator = op;

**if** (op=='+'){

value = left + right;

}**else** {

value = left - right;

}

}

// 实际产生对象的三个方法

**public** BinaryOperation\_3\_1 generateAdditionOperation() {

Random random = **new** Random();

**int** left, right, result;

left = random.nextInt(***UPPER***+1);

**do** {

right = random.nextInt(***UPPER***+1);

result = left + right;

} **while** (result > ***UPPER***);

BinaryOperation\_3\_1 bop = **new** BinaryOperation\_3\_1();

bop.construct(left, right, '+');

**return** bop;

}

**public** BinaryOperation\_3\_1 generateSubstractOperation(){

Random random = **new** Random();

**int** left, right, result;

left = random.nextInt(***UPPER***+1);

**do** {

right = random.nextInt(***UPPER***+1);

result = left - right;

} **while** (result < 0);

BinaryOperation\_3\_1 bop = **new** BinaryOperation\_3\_1();

bop.construct(left, right, '-');

**return** bop;

}

**public** BinaryOperation\_3\_1 generateBinaryOperation() {

Random random = **new** Random();

**int** opValue = random.nextInt(2);

**if** (opValue == 1){

**return** generateAdditionOperation();

} **else** {

**return** generateSubstractOperation();

}

}

// 实例变量访问器

**public** **int** getLeftOperand(){**return** left\_operand;}

**public** **int** getRightOperand(){**return** right\_operand;}

**public** **char** getOperator(){**return** operator;}

**public** **int** getResult(){**return** value;}

**public** **boolean** equals (BinaryOperation\_3\_1 anOperation) { // 要使用 getOperator()

**return** left\_operand == anOperation.getLeftOperand() &

right\_operand == anOperation.getRightOperand() &

operator == anOperation.getOperator();

}

**public** String toString(){

**return** ""+left\_operand+getOperator()+right\_operand;

}

**public** String asString(){

**return** toString()+"=";

}

**public** String fullString(){

**return** toString()+"="+getResult();

}

}

**package** cbsc.cha3;

**public** **class** Exercise\_3\_1 {

**private** **static** **final** **short** ***OPERATION\_NUMBER***=50;

**private** **static** **final** **short** ***COLUMN\_NUMBER***=5;

**private** BinaryOperation\_3\_1 operationList[] = **new** BinaryOperation\_3\_1 [***OPERATION\_NUMBER***];

// 在数组中增加算术运算题前先检查是否已经在数组中，以避免重复，

**public** **void** generateBinaryExercise() {

BinaryOperation\_3\_1 anOperation, opCreator = **new** BinaryOperation\_3\_1();

**for**(**int** i=0; i < ***OPERATION\_NUMBER***; i++){

anOperation = opCreator.generateBinaryOperation();

**while** (contains(anOperation,i-1)){

anOperation = opCreator.generateBinaryOperation();

}

operationList[i]= anOperation;

}

}

**public** **void** generateAdditionExercise(){

BinaryOperation\_3\_1 anOperation, opCreator = **new** BinaryOperation\_3\_1();

**for**(**int** i=0; i < ***OPERATION\_NUMBER***; i++){

anOperation = opCreator.generateAdditionOperation();

**while** (contains(anOperation,i-1)){

anOperation = opCreator.generateAdditionOperation();

}

operationList[i]= anOperation;

}

}

**public** **void** generateSubstractExercise(){

BinaryOperation\_3\_1 anOperation, opCreator = **new** BinaryOperation\_3\_1();

**for**(**int** i=0; i < ***OPERATION\_NUMBER***; i++){

anOperation = opCreator.generateSubstractOperation();

**while** (contains(anOperation,i-1)){

anOperation = opCreator.generateSubstractOperation();

}

operationList[i]= anOperation;

}

}

// 只要产生的算式没在当前练习题中，就加入尾部

**private** **boolean** contains (BinaryOperation\_3\_1 anOperation, **int** length){

**boolean** found=**false**;

**for**(**int** i=0; i <= length; i++) {

// System.out.println("--"+i+"--");

**if** (anOperation.equals(operationList[i])){

found = **true**;

**break**;

}

}

**return** found;

}

**void** formateAndDisplay (){

**int** count=1;

**for**(**int** i=0; i < ***OPERATION\_NUMBER***; i++) {

**if** (count > ***COLUMN\_NUMBER***){

*print*("\n");

count = 1;

}

*print*(""+(i+1)+"."+" ");

*print*(operationList[i].asString()+"\t");

count++;

}

*print*("\n");

}

**void** printAll(**int** count){

**for**(**int** i=0; i < count; i++) {

*print*(operationList[i].fullString());

*print*(", ");

}

System.***out***.println();

}

**void** printAll(){

**for**(**int** i=0; i < ***OPERATION\_NUMBER***; i++) {

*print*(operationList[i].asString());

*print*(", ");

}

*print*("\n");

}

**private** **static** **void** print(String str){

System.***out***.print(str);

}

**public** **static** **void** main(String[] args) {

// 产生一个Exercise对象，内容是空

Exercise\_3\_1 anExercise = **new** Exercise\_3\_1();

System.***out***.println("屏幕显示50道加法运算题：");

anExercise.generateAdditionExercise();

anExercise.formateAndDisplay();

System.***out***.println("屏幕显示50道减法运算题：");

anExercise.generateSubstractExercise();

anExercise.formateAndDisplay();

anExercise.generateBinaryExercise();

System.***out***.println("屏幕显示50道加法或减法运算题：");

anExercise.formateAndDisplay();

}

}

**package** cbsc.cha3;

**public** **class** BinaryOperationTester {

**public** **static** **void** main(String[] args) {

// 设计一：只有一个类，设计二：有两个子类

BinaryOperation\_3\_2 bop;

System.***out***.println("test1:constructor('+')");

**for** (**int** i=0; i<10; i++){

bop = **new** AdditionOperation();

System.***out***.println(bop);

}

System.***out***.println("test1:constructor('-')");

**for** (**int** i=0; i<10; i++){

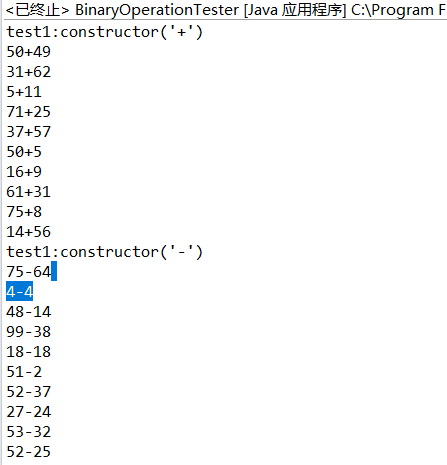
bop = **new** SubstractOperation();

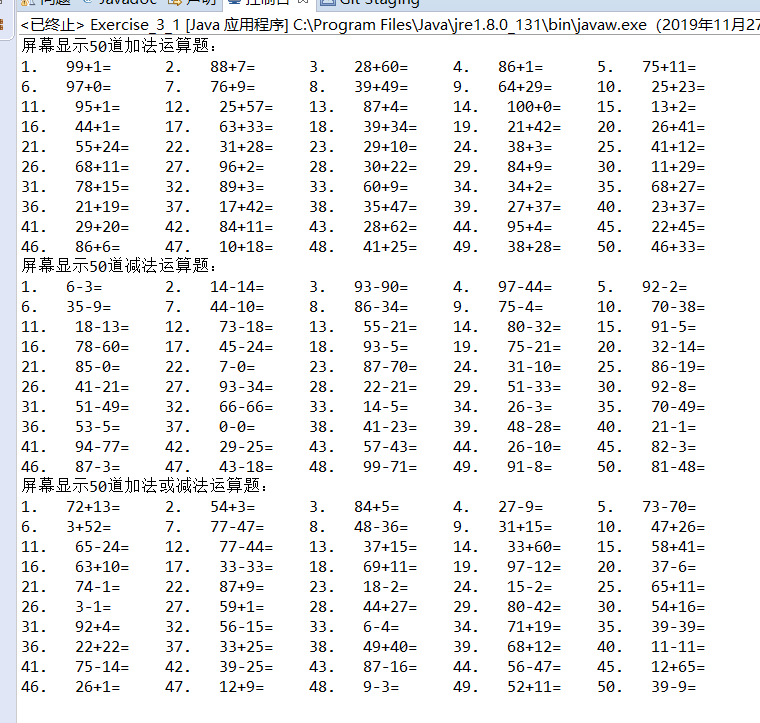
System.***out***.println(bop);

}

}

}





2.

package cbsc.cha3;

// 设计二

// 使用者（如Exercise）必须决定是选择加法运算还是减法运算，不能像之前一样，含糊其辞

// 因为BinaryOperation是抽象类，不能实例化

import java.util.Random;

public abstract class BinaryOperation\_3\_2 {

static final int UPPER = 100;

static final int LOWER = 0;

private int left\_operand=0, right\_operand=0;

private char operator='+';

private int value=0;

protected void generateBinaryOperation(char anOperator) {

int left, right, result;

Random random = new Random();

left = random.nextInt(UPPER+1);

do {

right = random.nextInt(UPPER+1);

result = calculate(left,right);

} while (!(checkingCalculation(result)));

left\_operand = left;

right\_operand = right;

operator = anOperator;

value = result;

}

// 2015-8-3. 只有当参数构成了有效的算式，才能调用

// 目前仅支持加法和减法

private void unsafeConstructor(int left,int right, char anOperator){

left\_operand = left;

right\_operand = right;

operator = anOperator;

value = anOperator == '+'?left+right:left-right;

}

public void unsafeConstructor(int left,int right, int result, char anOperator){

left\_operand = left;

right\_operand = right;

operator = anOperator;

value = result;

}

public void unsafeConstructor(String eqString){

int opPos=0;

int length=eqString.length();

// try to locate the position of the operator either '+' or '-'

opPos=eqString.indexOf("+");

if (opPos <= 0){

opPos=eqString.indexOf("-");

}

unsafeConstructor(Integer.parseInt(eqString.substring(0,opPos)),

Integer.parseInt(eqString.substring(opPos+1,length)),

eqString.charAt(opPos));

}

abstract boolean checkingCalculation(int anInteger);

abstract int calculate(int left, int right);

public int getLeftOperand(){return left\_operand;}

public int getRightOperand(){return right\_operand;}

public char getOperator(){return operator;}

public int getResult(){return value;}

public boolean equals (BinaryOperation\_3\_2 anOperation) { // 要使用 getOperator()

return left\_operand == anOperation.getLeftOperand() &

right\_operand == anOperation.getRightOperand() &

operator == anOperation.getOperator();

}

public String toString(){

return ""+left\_operand+getOperator()+right\_operand;

}

public String asString(){

return toString()+"=";

}

public String fullString(){

return toString()+"="+getResult();

}

}

package cbsc.cha3;

import java.io.File;

import java.util.Iterator;

public class ExerciseSheet\_3\_3\_3 {

private static final short COLUMN\_NUMBER=5;

public void formattedDisplay (Exercise\_3\_2\_3 ex, int columns){

Iterator<BinaryOperation\_3\_2> iterator = ex.iterator();

// 0 < columns

int column=1;

int count = 1;

while(iterator.hasNext()){

if (column > columns){

print("\n");

column = 1;

}

print(""+count+"."+" ");

print((iterator.next()).asString()+"\t");

count++;

column++;

}

print("\n");

}

public void formattedDisplay2 (Exercise\_3\_2\_3 ex, int columns){

// Iterator<BinaryOperation\_3\_2> iterator = ex.iterator();

int column=1;

int count = 1;

while(ex.hasNext()){

if (column > columns){

print("\n");

column = 1;

}

print(""+count+"."+" ");

print((ex.next()).asString()+"\t");

count++;

column++;

}

print("\n");

}

public void formattedDisplay (Exercise\_3\_2\_3 ex){

formattedDisplay (ex,COLUMN\_NUMBER);

}

private static void print(String str){

System.out.print(str);

}

public static void main(String[] args) {

ExerciseSheet\_3\_3\_3 sheet = new ExerciseSheet\_3\_3\_3();

Exercise\_3\_2\_3 exercise = new Exercise\_3\_2\_3();

Exercise\_3\_2\_3 ex = new Exercise\_3\_2\_3();

exercise.generateAdditionExercise(26);

System.out.println("---- generate and display add exercises ----");

sheet.formattedDisplay2(exercise,4);

System.out.println("---- generate and display exercises ----");

exercise.generateSubstractExercise(20);

sheet.formattedDisplay2(exercise,3);

System.out.println("---- read exercises in a file ----");

ex=exercise.readCSVExercise(new File("test2.txt"));

sheet.formattedDisplay(ex,4);

ex.writeResults(new File("results.txt"));

}

}

**package** cbsc.cha3;

**public** **class** SubstractOperation **extends** BinaryOperation\_3\_2 {

**public** SubstractOperation() {

generateBinaryOperation('-');

}

**boolean** checkingCalculation(**int** anInteger){

**return** anInteger >= ***LOWER***;

}

**int** calculate(**int** left, **int** right){

**return** left-right;

}

}

**package** cbsc.cha3;

//

**public** **class** AdditionOperation **extends** BinaryOperation\_3\_2 {

**public** AdditionOperation() {

generateBinaryOperation('+');

}

**public** **boolean** checkingCalculation(**int** anInteger){

**return** anInteger <= ***UPPER***;

}

**int** calculate(**int** left, **int** right){

**return** left+right;

}

}

**package** cbsc.cha3;

**import** java.io.\*;

**import** java.util.Iterator;

**import** java.util.Random;

**import** java.util.ArrayList;

**import** java.util.Scanner;

**public** **class** Exercise\_3\_2\_3 {

**private** ArrayList<BinaryOperation\_3\_2> operationList = **new** ArrayList<BinaryOperation\_3\_2>();

**private** **int** current=0; // only used for iterator

**private** BinaryOperation\_3\_2 generateOperation(){

Random random = **new** Random();

**int** opValue = random.nextInt(2);

**if** (opValue == 1){

**return** **new** AdditionOperation();

}

**return** **new** SubstractOperation();

}

**public** **void** generateAdditionExercise( **int** operationCount){

BinaryOperation\_3\_2 anOperation;

**while** (operationCount > 0 ){

**do** {anOperation = **new** AdditionOperation();

}**while** (operationList.contains(anOperation));

operationList.add(anOperation);

// System.out.println("count="+ operationList.size());

operationCount--;

}

}

**public** **void** generateBinaryExercise(**int** operationCount){

BinaryOperation\_3\_2 anOperation;

**while** (operationCount > 0 ){

**do**{anOperation = generateOperation();

}**while** (operationList.contains(anOperation));

operationList.add(anOperation);

operationCount--;

}

}

**public** **void** generateSubstractExercise(**int** operationCount){

BinaryOperation\_3\_2 anOperation;

**while** (operationCount > 0 ){

**do**{anOperation = **new** SubstractOperation();

}**while** (operationList.contains(anOperation));

operationList.add(anOperation);

operationCount--;

}

}

**public** **void** add(BinaryOperation\_3\_2 anOperation){

operationList.add(anOperation);

}

**public** **boolean** contains(BinaryOperation\_3\_2 anOperation){

**return** operationList.contains(anOperation);

}

**public** **int** length(){

**return** operationList.size();

}

**public** **void** writeExercise(){

File wfile = **new** File("eq2.txt");

**try**{

Writer out = **new** FileWriter(wfile, **true**);

**for** (BinaryOperation\_3\_2 op: operationList){

out.write(op.toString()+",");

}

out.flush();

out.close();

}

**catch**(IOException e){

System.***out***.println("ERROR: "+e);

}

}

**public** **void** writeCSVExercise(File aFile){

**try**{

Writer out = **new** FileWriter(aFile, **true**);

**for** (BinaryOperation\_3\_2 op: operationList){

out.write(op.toString()+",");

}

out.flush();

out.close();

}

**catch**(IOException e){

System.***out***.println("ERROR: "+e);

}

}

**public** Exercise\_3\_2\_3 readCSVExercise(){

Exercise\_3\_2\_3 exercise = **new** Exercise\_3\_2\_3();

String eqString;

BinaryOperation\_3\_2 op;

Scanner sc = **null**;

File rfile = **new** File("eq2.txt");

**try**{

sc = **new** Scanner(rfile);

sc.useDelimiter(",\\n");

**while**(sc.hasNext()){

eqString = sc.next();

op = **new** AdditionOperation();

op.unsafeConstructor(eqString);

exercise.add(op);

}

}

**catch**(IOException e){

System.***out***.println("ERROR: "+e);

}

**return** exercise;

}

**public** Exercise\_3\_2\_3 readCSVExercise(File aFile){

Exercise\_3\_2\_3 exercise = **new** Exercise\_3\_2\_3();

String eqString;

BinaryOperation\_3\_2 op;

**try**{

**new** Scanner(aFile).useDelimiter(",");

**while**(**new** Scanner(aFile).hasNext()){

eqString = **new** Scanner(aFile).next().replaceAll("\\s", "");

op = **new** AdditionOperation();

op.unsafeConstructor(eqString);

exercise.add(op);

}

}

**catch**(IOException e){

System.***out***.println("ERROR: "+e);

}

**return** exercise;

}

**public** **boolean** hasNext(){

**return** current <= operationList.size()-1;

}

**public** BinaryOperation\_3\_2 next(){

**return** operationList.get(current++);

}

**public** **void** printCurrent(){

System.***out***.println("current="+current);

}

**public** **void** all(){

**for** (BinaryOperation\_3\_2 op:operationList){

System.***out***.println(op.asString());

}

}

**public** **void** writeResults(File aFile){

**try**{

Writer out = **new** FileWriter(aFile, **true**);

**for** (BinaryOperation\_3\_2 op: operationList){

out.write(op.getResult()+",");

}

out.flush();

out.close();

}

**catch**(IOException e){

System.***out***.println("ERROR: "+e);

}

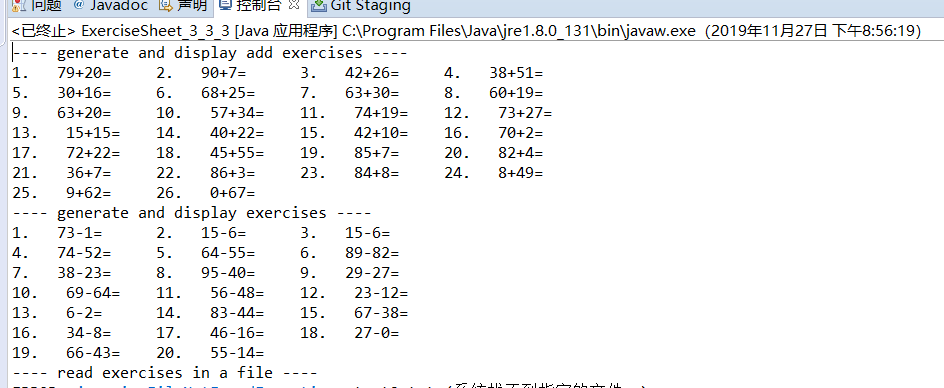
}

**public** Iterator<BinaryOperation\_3\_2> iterator(){

**return** operationList.iterator();

}

}



**import** java.util.Random;

**public** **abstract** **class** Method\_random {

**private** **int** n;

**public** Method\_random() {

}

// 产生加减用到的随机数

**public** **int** TheRandomNumber( ){

Random random=**new** Random();

n=(**short**)random.nextInt(101);

**return** n;

}

// 产生加减用到的随机数存到数组中

**public** **abstract** **void** TheRandomNumber2(**int** a[],**int** b[]);

//决定算式是加法或减法

**public** **abstract** **void** Formula(**char**[] m );

}

**public** **class** Mathod\_random2 **extends** Method\_random{

**private** **int** i,k;

// 产生加减用到的随机数存到数组中

**public** **void** TheRandomNumber2(**int** a[],**int** b[])

{

**for**(i=0;i<50;i++)

{

a[i]=TheRandomNumber( );

b[i]=TheRandomNumber( );

}

a[i]='\0';

b[i]='\0';

}

//决定算式是加法或减法

**public** **void** Formula(**char**[] m )

{

**for**(i=0;i<50;i++)

{

k=TheRandomNumber( )%2;

**if**(k==0)//随机数是偶数即为加算法，奇数为减法

m[i]='+';

**else**

m[i]='-';

}

m[i]='\0';

}

}

**public** **class** Method\_arithmetic {

**private** **int** i,n;

//产生完整算式

**public** **void** Arithmetic(**int** a[],**int** b[],**char** c[])

{

n=0;

**for**(i=0;i<50;i++)

{

System.***out***.print(" "+a[i]+c[i]+b[i]+"=");

n=n+1;

**if**(n%5==0)

System.***out***.println();

}

}

}

**public** **class** Method\_check {

//检查加法算式的和不能超过100，减法算式的差不能小于0

**private** **int** i,j;

Method\_random r=**new** Mathod\_random2 ();

**public** **int** check(**int** a[],**int** b[])

{

**for**(i=0;i<50;i++)

{

**if**(a[i]+b[i]>100 || a[i]-b[i]<0)

{

a[i]=r.TheRandomNumber( );

b[i]=r.TheRandomNumber( );

check(a,b);

}

}

**return** 0;

}

//检查是否有相同的算式

**public** **void** Repeat(**int** a[],**int** b[],**char** c[])

{

**for**(i=0;i<50;i++)

**for**(j=0;j<50;j++)

**if**(a[i]==a[j])

**if**(b[i]==b[j])

**if**(c[i]==c[j])

a[j]=r.TheRandomNumber( );

}

}

**public** **class** Method\_result {

**private** **int** i,n;

//计算算式结果

**public** **void** Result(**int** a[],**int** b[],**char** c[])

{

n=0;

**for**(i=0;i<50;i++)

{

**if**(c[i]=='+')

System.***out***.print(" "+a[i]+c[i]+b[i]+"="+(a[i]+b[i]));

**else**

System.***out***.print(" "+a[i]+c[i]+b[i]+"="+(a[i]-b[i]));

n=n+1;

**if**(n%5==0)

System.***out***.println();

}

}

}

**import** java.util.Scanner;

**public** **class** Test\_again {

**public** **static** **void** main(String[] args) {

**int** a[]=**new** **int**[60];

**int** b[]=**new** **int**[60];

**char** c[]=**new** **char**[60];

Method\_random random=**new** Mathod\_random2();

Method\_arithmetic arithmetic=**new** Method\_arithmetic();

Method\_check check=**new** Method\_check();

Method\_result result=**new** Method\_result();

random.TheRandomNumber2(a,b);//产生加减用到的随机数存到数组中

check.Repeat(a,b,c);//检查是否有相同的算式

check.check(a,b);//检查加法算式的和不能超过100，减法算式的差不能小于0

random.Formula(c);//决定算式是加法或减法

arithmetic.Arithmetic(a,b,c);//产生完整算式

System.***out***.print("是否输出各个算式的结果,是则输入‘1’\n");

Scanner in =**new** Scanner(System.***in***);

**int** d1=in.nextInt();

**if**(d1==1)

result.Result(a,b,c);//计算算式结果

}

}

**五、分析与思考**

if(c[i]=='+')

System.out.print(" "+a[i]+c[i]+b[i]+"="+(a[i]+b[i]));

else

System.out.print(" "+a[i]+c[i]+b[i]+"="+(a[i]-b[i]));

n=n+1;

if(n%5==0)

System.out.println();实验时要注意这个循环，仔细看

**六、实验总结**

感觉好复杂，几近崩溃，还是要继续努力的。