数卡源码

2015-07-20 陈云龙

# 摘要：

此代码采用java编写，实现图形化界面，ui部分代码与算法无关，故不提供，以下是包结构和源码。

# main/Launcher.java

|  |
| --- |
| package main;  import java.awt.event.MouseAdapter;  import java.awt.event.MouseEvent;  import java.text.SimpleDateFormat;  import java.util.Arrays;  import java.util.Date;  import cardbl.Card;  import cardbl.Create;  import cardbl.CreateFive;  import cardbl.CreateFour;  import cardbl.CreateThree;  import ui.UI;  /\*\*  \* 数卡设计  \* **@author** cylong  \* **@version** 2015年7月17日 下午6:03:02  \*/  public class Launcher {    public static UI *ui* = new UI();  public static final Date *CURRENT\_TIME* = new Date(); // 当前日期，当作文件名  public static final SimpleDateFormat *FORMATTER* = new SimpleDateFormat("yyyyMMddHHmmss");    public static int *CARD\_NUM*; // 卡片数量  public static final int *STATE* = 3; // 使用正、反面或者不使用    public static Card[][] *cardList*; // 一套卡片    public static int *MIN\_NUM* = 1; // 最小值，就是1  public static int *MAX\_NUM*; // 识别量  public static int *LOWER\_R*; // R的下界  public static int *MIN\_R*; // 计算出来的最小R  public static int *R* = 0; // 想要输出的容量为r的数卡      //---------------------- 三张卡片的基础数------------------  public static int[][] *firstNum\_three* = {{0, 1, 2},  {0, 3, 0},  {0, 0, 0}};    public static int[][] *range\_three* = {{4, 6},  {6, 9},  {10, 18}};  //---------------------- 三张卡片的基础数------------------      //---------------------- 四张卡片的基础数------------------  public static int[][] *firstNum\_four* = {{0, 1, 2},  {0, 3, 0},  {0, 0, 0},  {0, 0, 0}};  public static int[][] *range\_four* = {{4, 6},  {6, 9},  {10, 18},  {14, 27},  {28, 54}};  //---------------------- 四张卡片的基础数------------------      //---------------------- 五张卡片的基础数------------------  public static int[][] *firstNum\_five* = {{0, 1, 2},  {0, 3, 6},  {0, 9, 0},  {0, 0, 0},  {0, 0, 0}};    public static int[][] *range\_five\_100* = {{16, 17},  {22, 25},  {32, 37},  {44, 51},  {56, 64}};    public static int[][] *range\_five\_1* = {{15, 17},  {22, 26},  {28, 42},  {36, 62},  {46, 75}};    public static int[][] *range\_five\_2* = {{18, 18},  {27, 27},  {36, 54},  {55, 81},  {67, 110}};    public static int[][] *range\_five\_3* = {{18, 18},  {27, 27},  {54, 54},  {70, 81},  {96, 162}};  //---------------------- 五张卡片的基础数------------------      public static void main(String[] args) {  *ui*.btn.addMouseListener(new MouseAdapter() {  @Override  public void mouseClicked(MouseEvent e) {  *ui*.setContent("R的最优结果计算中...\r\n");  new Thread() {  public void run() {  Create.*minR* = 1000;  *MAX\_NUM* = Integer.*parseInt*(*ui*.getInput());  String out = *startCalc*();  *ui*.setContent(out);  };  }.start();  }  });  }    private static synchronized String startCalc() {  String out = "";  if(*MAX\_NUM* <= 0) {  out += "请输入大于0的数\r\n";  return out;  }  out += "R的最优结果计算中...\r\n";  if(*MAX\_NUM* <= 8) { // 前8个识别量直接输出  *createCard*(*MAX\_NUM*);  } else {  *CARD\_NUM* = *getCardNum*(*MAX\_NUM*); // 求卡片数量  *LOWER\_R* = *getLowerR*(*MAX\_NUM*); // 求r的下界  if(*CARD\_NUM* == 3) { // 3张卡片的数卡  *startCalcThree*(*firstNum\_three*, *range\_three*);  } else if(*CARD\_NUM* == 4) { // 4张卡片的数卡  *startCalcFour*(*firstNum\_four*, *range\_four*);  } else if(*CARD\_NUM* == 5) { // 5张卡片的数卡  if(*MAX\_NUM* == 100) { // 题目中的数字，单独计算  *startCalcFive*(*firstNum\_five*, *range\_five\_100*);  } else if(*MAX\_NUM* >= 81 && *MAX\_NUM* <= 120) {  *startCalcFive*(*firstNum\_five*, *range\_five\_1*);  } else if(*MAX\_NUM* >= 121 && *MAX\_NUM* <= 161) {  *startCalcFive*(*firstNum\_five*, *range\_five\_2*);  } else if(*MAX\_NUM* >= 162 && *MAX\_NUM* <= 242) {  *startCalcFive*(*firstNum\_five*, *range\_five\_3*);  }  } else {  out += "输入数字范围不正确\r\n";  return out;  }  *MIN\_R* = Create.*minR*;  *cardList* = Create.*minCardList*;  }    out += ("卡片数量 = " + *CARD\_NUM* + "\r\n");  out += ("R的下界 = " + *LOWER\_R* + "\r\n");  out += ("R最优结果 = " + *MIN\_R* + "\r\n");  out += "\r\n数卡: \r\n";  int sum = 0;  for(int i = 0; i < *cardList*.length; i++) {  for(int j = 1; j < *cardList*[i].length; j++) {  if(*cardList*[i][j].index == 1 && *cardList*[i][j].numList[0] > *MAX\_NUM*) {  *cardList*[i][j].index = 0;  }  sum += *cardList*[i][j].index \* *cardList*[i][j].numList[0];  out += ("卡" + (i + 1) + (j == 1 ? "正面" : "反面") + " 数字个数: " + *cardList*[i][j].index + "\r\n");  Arrays.*sort*(*cardList*[i][j].numList, 0, *cardList*[i][j].index);  for(int k = 0; k < *cardList*[i][j].index; k++) {  out += (*cardList*[i][j].numList[k] + " ");  }  out += "\r\n\r\n";  }  }  out += ("Sum = " + sum + "\r\n");  return out;  }    public static void createCard(int n) {  if(n <= 2) {  *CARD\_NUM* = 1;  *LOWER\_R* = 1;  *MIN\_R* = 1;  *cardList* = new Card[Launcher.*CARD\_NUM*][Launcher.*STATE*];  for(int i = 0; i < *cardList*.length; i++) {  for(int j = 0; j < *cardList*[i].length; j++) {  *cardList*[i][j] = new Card();  }  }  switch(n) {  case 1:  *cardList*[0][1].numList[0] = 1;  *cardList*[0][1].index = 1;    break;  case 2:  *cardList*[0][1].numList[0] = 1;  *cardList*[0][1].index = 1;    *cardList*[0][2].numList[0] = 2;  *cardList*[0][2].index = 1;  break;  }  } else {  *CARD\_NUM* = 2;  *cardList* = new Card[Launcher.*CARD\_NUM*][Launcher.*STATE*];  for(int i = 0; i < *cardList*.length; i++) {  for(int j = 0; j < *cardList*[i].length; j++) {  *cardList*[i][j] = new Card();  }  }  switch(n) {  case 3:  *cardList*[0][1].numList[0] = 1;  *cardList*[0][1].index = 1;    *cardList*[0][2].numList[0] = 2;  *cardList*[0][2].index = 1;    *cardList*[1][1].numList[0] = 3;  *cardList*[1][1].index = 1;    *LOWER\_R* = 1;  *MIN\_R* = 1;  break;  case 4:  *cardList*[0][1].numList[0] = 1;  *cardList*[0][1].index = 1;    *cardList*[0][2].numList[0] = 2;  *cardList*[0][2].index = 1;    *cardList*[1][1].numList[0] = 3;  *cardList*[1][1].index = 1;    *cardList*[1][2].numList[0] = 4;  *cardList*[1][2].index = 1;    *LOWER\_R* = 1;  *MIN\_R* = 1;  break;  case 5:  *cardList*[0][1].numList[0] = 1;  *cardList*[0][1].index = 1;    *cardList*[0][2].numList[0] = 2;  *cardList*[0][2].numList[1] = 5;  *cardList*[0][2].index = 2;    *cardList*[1][1].numList[0] = 3;  *cardList*[1][1].numList[1] = 5;  *cardList*[1][1].index = 2;    *cardList*[1][2].numList[0] = 4;  *cardList*[1][2].index = 1;    *LOWER\_R* = 2;  *MIN\_R* = 2;  break;  case 6:  *cardList*[0][1].numList[0] = 1;  *cardList*[0][1].numList[1] = 4;  *cardList*[0][1].index = 2;    *cardList*[0][2].numList[0] = 2;  *cardList*[0][2].numList[1] = 5;  *cardList*[0][2].index = 2;    *cardList*[1][1].numList[0] = 3;  *cardList*[1][1].numList[1] = 4;  *cardList*[1][1].numList[2] = 5;  *cardList*[1][1].index = 3;    *cardList*[1][2].numList[0] = 6;  *cardList*[1][2].index = 1;    *LOWER\_R* = 3;  *MIN\_R* = 3;  break;  case 7:  *cardList*[0][1].numList[0] = 1;  *cardList*[0][1].numList[1] = 4;  *cardList*[0][1].numList[2] = 7;  *cardList*[0][1].index = 3;    *cardList*[0][2].numList[0] = 2;  *cardList*[0][2].numList[1] = 5;  *cardList*[0][2].index = 2;    *cardList*[1][1].numList[0] = 3;  *cardList*[1][1].numList[1] = 4;  *cardList*[1][1].numList[2] = 5;  *cardList*[1][1].index = 3;    *cardList*[1][2].numList[0] = 6;  *cardList*[1][2].numList[1] = 7;  *cardList*[1][2].index = 2;    *LOWER\_R* = 3;  *MIN\_R* = 3;  break;  case 8:  *cardList*[0][1].numList[0] = 1;  *cardList*[0][1].numList[1] = 4;  *cardList*[0][1].numList[2] = 7;  *cardList*[0][1].index = 3;    *cardList*[0][2].numList[0] = 2;  *cardList*[0][2].numList[1] = 5;  *cardList*[0][2].numList[2] = 8;  *cardList*[0][2].index = 3;    *cardList*[1][1].numList[0] = 3;  *cardList*[1][1].numList[1] = 4;  *cardList*[1][1].numList[2] = 5;  *cardList*[1][1].index = 3;    *cardList*[1][2].numList[0] = 6;  *cardList*[1][2].numList[1] = 7;  *cardList*[1][2].numList[2] = 8;  *cardList*[1][2].index = 3;    *LOWER\_R* = 3;  *MIN\_R* = 3;  break;  default:  break;  }    }    }  private static void startCalcThree(int[][] firstNum, int[][] range) {  for(int i = range[0][0]; i <= range[0][1]; i++) {  for(int j = range[1][0]; j <= range[1][1]; j++) {  for(int k = range[2][0]; k <= range[2][1]; k++) {  if(*isUp*(i, j, k)) {  firstNum[1][2] = i;  firstNum[2][1] = j;  firstNum[2][2] = k;  CreateThree createCard = new CreateThree();  createCard.start(firstNum);  if(Create.*minR* == *LOWER\_R*) {  return; // 找到一个最小r就好  }  }  }  }  }  }    private static void startCalcFour(int[][] firstNum, int[][] range) {  for(int i = range[0][0]; i <= range[0][1]; i++) {  for(int j = range[1][0]; j <= range[1][1]; j++) {  for(int k = range[2][0]; k <= range[2][1]; k++) {  for(int m = range[3][0]; m <= range[3][1]; m++) {  for(int n = range[4][0]; n <= range[4][1]; n++) {  if(*isUp*(i, j, k, m, n)) {  firstNum[1][2] = i;  firstNum[2][1] = j;  firstNum[2][2] = k;  firstNum[3][1] = m;  firstNum[3][2] = n;  CreateFour createCard = new CreateFour();  createCard.start(firstNum);  if(Create.*minR* == *LOWER\_R*) {  return; // 找到一个最小r就好  }  }  }  }  }  }  }  }    private static void startCalcFive(int[][] firstNum, int[][] range) {  for(int i = range[0][0]; i <= range[0][1]; i++) {  for(int j = range[1][0]; j <= range[1][1]; j++) {  for(int k = range[2][0]; k <= range[2][1]; k++) {  for(int m = range[3][0]; m <= range[3][1]; m++) {  for(int n = range[4][0]; n <= range[4][1]; n++) {  if(*isUp*(i, j, k, m, n)) {  firstNum[2][2] = i;  firstNum[3][1] = j;  firstNum[3][2] = k;  firstNum[4][1] = m;  firstNum[4][2] = n;  CreateFive createCard = new CreateFive();  createCard.start(firstNum);  if(Create.*minR* == *LOWER\_R*) {  return; // 找到一个最小r就好  }  }  }  }  }  }  }  }    /\*\*  \* **@author** cylong  \* **@version** 2015年7月19日 下午7:13:07  \* **@return**  \*/  private static boolean isUp(int...args) {  for(int i = 1; i < args.length; i++) {  if(args[i - 1] >= args[i]) {  return false;  }  }  return true;  }  private static int getCardNum(int n) {  double d\_cardNum = Math.*log*(n + 1) / Math.*log*(*STATE*);  int i\_cardName = (int)d\_cardNum;  if(i\_cardName == d\_cardNum) {  return i\_cardName;  } else {  return i\_cardName + 1;  }  }    /\*\*  \* **@author** cylong  \* **@version** 2015年7月19日 下午5:32:11  \*/  private static int getLowerR(int n) {  int k = *getCardNum*(n);  int[] arr = new int[k];  arr[0] = 2 \* *Cnr*(k, 1);  int t = 0;  for(int i = 1; i < arr.length; i++) {  arr[i] = (int)(Math.*pow*(2, i + 1) \* *Cnr*(k, i + 1) + arr[i - 1]);  if(n >= arr[i - 1] && n <= arr[i]) {  t = i - 1;  }  }    int a = n - arr[t];  int s = 0;  for(int i = 1; i <= t + 1; i++) {  s += Math.*pow*(2, i) \* *Cnr*(k, i) \* i;  }  s += a \* (t + 2);  double d\_r = s \* 1.0 / (2 \* k);  int i\_r = (int)d\_r;  if(i\_r == d\_r) {  int m = n \* (n + 1) / 2;  if(m % i\_r == 0) {  return i\_r;  } else {  return i\_r + 1;  }  } else {  return i\_r + 1;  }  }    private static int Cnr(int n, int r) {  int pro = 1;  for(int i = n - r + 1; i <= n; i++) {  pro \*= i;  }  int res = pro / *factorial*(r);  return res;  }    private static int factorial(int n) {  if(n < 0) {  return -1;  } else if(n == 0) {  return 1;  } else {  return n \* *factorial*(n - 1);  }  }    } |

# cardbl/Card.java

|  |
| --- |
| package cardbl;  /\*\*  一面的卡片  **@author** cylong  **@version** 2015年7月17日 下午8:06:54  \*/  public class Card {  public int index = 0; // 下面数组的下标  public int numList[] = new int[100]; // 每一面存储的数字  } |

# cardbl/Create.java

|  |
| --- |
| package cardbl;  import java.io.BufferedWriter;  import java.io.File;  import java.io.FileWriter;  import java.io.IOException;  import java.util.Arrays;  import main.Launcher;  /\*\*  \* 卡片的计算  \* **@author** cylong  \* **@version** 2015年7月17日 下午8:09:42  \*/  public abstract class Create {  /\*\* 数字范围 \*/  protected int[] numRange = new int[Launcher.*MAX\_NUM* - Launcher.*MIN\_NUM* + 1];  /\*\* 范围内数字是否命中 \*/  protected boolean[] numExist = new boolean[Launcher.*MAX\_NUM* - Launcher.*MIN\_NUM* + 1];  /\*\* 生成的卡片 \*/  public Card[][] cardList = new Card[Launcher.*CARD\_NUM*][Launcher.*STATE*];  public static int *minR* = 10000;  /\*\* 最小R的卡片 \*/  public static Card[][] *minCardList*;  /\*\* 初始的10个基础数 \*/  protected int[][] firstNum;  public int start(int[][] firstNum) {  this.firstNum = firstNum;  for(int i = 0; i < numRange.length; i++) {  numRange[i] = i + Launcher.*MIN\_NUM*;  }  for(int i = 0; i < cardList.length; i++) {  for(int j = 0; j < cardList[i].length; j++) {  cardList[i][j] = new Card();  cardList[i][j].numList[0] = firstNum[i][j];  cardList[i][j].index++;  numIsExist(firstNum[i][j]); // 初始化的10个数字出现过  }  }  startLoop();  return *minR*;  }  protected void startLoop() {  for(int i = 0; i < 15; i++) {  calc();  int r = findR();  if (r != -1) {  if (r == Launcher.*R*) {  *print*(cardList);  writeToFile();  }  } else {  break;  }  }  }  protected abstract void calc();    protected void handle(int...args) {  int sum = 0;  for(int i = 0; i < args.length; i++) {  sum += cardList[i][args[i]].numList[0];  }  if (isFirst(sum)) {  return;  }  int temp = numIsExist(sum);  if (temp == 2) {  Card[][] newCardList = copyArray(this.cardList);  newCardList = remove(newCardList, sum);  addToCardList(newCardList, sum, args);  int curTmpR = getTempR(this.cardList);  int newTmpR = getTempR(newCardList);  int curNum = getTotalNum(this.cardList);  int newNum = getTotalNum(newCardList);  double curVar = getVar(this.cardList);  double newVar = getVar(newCardList);  if (newTmpR < curTmpR) { // 取R小的  this.cardList = newCardList;  return;  } else if (newTmpR == curTmpR) {  if (newNum < curNum) { // 取数字数量小的  this.cardList = newCardList;  return;  } else if (newNum == curNum) {  if (newVar < curVar) { // 取方差小的  this.cardList = newCardList;  return;  }  }  }  }  if (temp == 1) {  addToCardList(cardList, sum, args);  }  }  /\*\* 求出来的和加入到卡片中 \*/  protected void addToCardList(Card[][] cardList, int sum, int...args) {  for(int i = 0; i < args.length; i++) {  if(args[i] != 0) {  cardList[i][args[i]].numList[cardList[i][args[i]].index++] = sum;  }  }  }  /\*\*  \* 获得卡片面中数字个数的方差  \* **@author** cylong  \* **@version** 2015年7月18日 下午9:08:24  \*/  protected double getVar(Card[][] cardList) {  int n = cardList.length \* (cardList[0].length - 1);  double avg = getTotalNum(cardList) \* 1.0 / n;  double varSum = 0.0;  for(int i = 0; i < cardList.length; i++) {  for(int j = 1; j < cardList[0].length; j++) {  varSum += Math.*pow*((cardList[i][j].index - avg), 2);  }  }  double var = varSum / n;  return var;  }  /\*\*  \* 获得卡片的数字个数  \* **@author** cylong  \* **@version** 2015年7月18日 下午8:22:38  \*/  protected int getTotalNum(Card[][] cardList) {  int sum = 0;  for(int i = 0; i < cardList.length; i++) {  for(int j = 1; j < cardList[i].length; j++) {  sum += cardList[i][j].index;  }  }  return sum;  }  /\*\* 获得当前的R \*/  protected int getTempR(Card[][] cardList) {  int maxR = 0;  for(int i = 0; i < cardList.length; i++) {  for(int j = 1; j < cardList[i].length; j++) {  maxR = maxR > cardList[i][j].index ? maxR : cardList[i][j].index;  }  }  return maxR;  }  /\*\*  \* 遇到重复的和，去除掉之前添加到卡片中的  \* **@author** cylong  \* **@version** 2015年7月17日 下午8:59:46  \*/  protected Card[][] remove(Card[][] newCardList, int sum) {  for(int i = 0; i < newCardList.length; i++) {  for(int j = 0; j < newCardList[i].length; j++) {  int numList[] = newCardList[i][j].numList;  for(int k = 0; k < numList.length; k++) {  if (numList[k] == sum) {  for(int m = k; m < numList.length - 1; m++) {  numList[m] = numList[m + 1];  }  newCardList[i][j].index--;  }  }  }  }  return newCardList;  }  /\*\* 计算卡片的R \*/  protected int findR() {  // 判断所有的数是否都出现过  for(int i = 0; i < numExist.length; i++) {  if (!numExist[i]) {  return -1;  }  }  int maxR = 0;  for(int i = 0; i < cardList.length; i++) {  for(int j = 1; j < cardList[i].length; j++) {  if (cardList[i][j].index > maxR) {  maxR = cardList[i][j].index;  }  }  }  if (*minR* > maxR) {  *minR* = maxR;  *minCardList* = cardList;  }  return maxR;  }  /\* 打印卡片 \*/  public static void print(Card[][] cardList) {  int sum = 0;  for(int i = 0; i < cardList.length; i++) {  for(int j = 1; j < cardList[i].length; j++) {  sum += cardList[i][j].index \* cardList[i][j].numList[0];  System.*out*.println("卡" + (i + 1) + (j == 1 ? "正面" : "反面") + " 数字个数: " + cardList[i][j].index);  Arrays.*sort*(cardList[i][j].numList, 0, cardList[i][j].index);  for(int k = 0; k < cardList[i][j].index; k++) {  System.*out*.print(cardList[i][j].numList[k] + " ");  }  System.*out*.println();  System.*out*.println();  }  }  System.*out*.println("Sum = " + sum);  }  /\*\* 写到文件中 \*/  protected void writeToFile() {  try {  String dateString = Launcher.*FORMATTER*.format(Launcher.*CURRENT\_TIME*);  String dirName = "data/" + Launcher.*MAX\_NUM* + "/" + dateString + "/";  String fileName = dirName + Launcher.*R*;  for(int i = 0; i < firstNum.length; i++) {  for(int j = 1; j < firstNum[i].length; j++) {  fileName += ("-" + firstNum[i][j]);  }  }  fileName += ".txt";  File dir = new File(dirName);  if (!dir.exists()) {  dir.mkdirs();  }  BufferedWriter bufWriter = new BufferedWriter(new FileWriter(fileName));  int sum = 0;  for(int i = 0; i < cardList.length; i++) {  for(int j = 1; j < cardList[i].length; j++) {  sum += cardList[i][j].index \* cardList[i][j].numList[0];  bufWriter.write("卡" + (i + 1) + (j == 1 ? "正面" : "反面") + " 数字个数: " + cardList[i][j].index);  bufWriter.newLine();  Arrays.*sort*(cardList[i][j].numList, 0, cardList[i][j].index);  for(int k = 0; k < cardList[i][j].index; k++) {  bufWriter.write(cardList[i][j].numList[k] + " ");  }  bufWriter.newLine();  bufWriter.newLine();  }  }  bufWriter.write("Sum = " + sum);  bufWriter.close();  } catch (IOException e) {  e.printStackTrace();  }  }  /\*\* 求出的和是否命中范围 \*/  protected int numIsExist(int num) {  for(int i = 0; i < numRange.length; i++) {  if (num == numRange[i]) {  if (numExist[i]) { // 这个数在之前已经出现过  return 2;  } else {  numExist[i] = true;  return 1;  }  }  }  return 0;  }  /\*\* 和是否是基础的数 \*/  protected boolean isFirst(int num) {  for(int i = 0; i < firstNum.length; i++) {  for(int j = 0; j < firstNum[i].length; j++) {  if (firstNum[i][j] == num) {  return true;  }  }  }  return false;  }  /\*\* 拷贝卡片 \*/  protected Card[][] copyArray(Card[][] cardList) {  Card[][] newCardList = new Card[cardList.length][cardList[0].length];  for(int i = 0; i < cardList.length; i++) {  for(int j = 0; j < cardList[i].length; j++) {  newCardList[i][j] = new Card();  newCardList[i][j].index = cardList[i][j].index;  newCardList[i][j].numList = cardList[i][j].numList.clone();  }  }  return newCardList;  }  } |

# cardbl/CreateFive.java

|  |
| --- |
| package cardbl;  import main.Launcher;  /\*\*  \* 5张卡片的计算  \* **@author** cylong  \* **@version** 2015年7月20日 上午1:50:25  \*/  public class CreateFive extends Create {    @Override  protected void calc() {  for(int i = Launcher.*STATE* - 1; i >= 0; i--) {  for(int j = Launcher.*STATE* - 1; j >= 0; j--) {  for(int k = Launcher.*STATE* - 1; k >= 0; k--) {  for(int m = Launcher.*STATE* - 1; m >= 0; m--) {  for(int n = Launcher.*STATE* - 1; n >= 0; n--) {  handle(i, j, k, m, n);  }  }  }  }  }  }  } |

# cardbl/CreateFour.java

|  |
| --- |
| package cardbl;  import main.Launcher;  /\*\*  \* 四张张卡片的计算  \* **@author** cylong  \* **@version** 2015年7月17日 下午8:09:42  \*/  public class CreateFour extends Create {  @Override  protected void calc() {  for(int i = Launcher.*STATE* - 1; i >= 0; i--) {  for(int j = Launcher.*STATE* - 1; j >= 0; j--) {  for(int k = Launcher.*STATE* - 1; k >= 0; k--) {  for(int m = Launcher.*STATE* - 1; m >= 0; m--) {  handle(i, j, k, m);  }  }  }  }  }  } |

# cardbl/CreateThree.java

|  |
| --- |
| package cardbl;  import main.Launcher;  /\*\*  \* 三张卡片的计算  \* **@author** cylong  \* **@version** 2015年7月20日 上午1:57:18  \*/  public class CreateThree extends Create {    @Override  protected void calc() {  for(int i = Launcher.*STATE* - 1; i >= 0; i--) {  for(int j = Launcher.*STATE* - 1; j >= 0; j--) {  for(int k = Launcher.*STATE* - 1; k >= 0; k--) {  handle(i, j, k);  }  }  }  }  } |

# ui

ui包中的内容是为了显示使用，与算法无关，故不提供