【案例6-4】 模拟订单号生成

### 【案例介绍】

**1.任务描述**

斗地主的扑克牌游戏，相信许多人都会玩，本例要求编写一个斗地主的洗牌发牌程序，要求按照斗地主的规则完成洗牌发牌的过程。一副扑克总共有54张牌，牌面由花色和数字组成（包括J、Q、K、A字母）组成，花色有♠、♥、♦、♣ 四种，分别表示黑桃、红桃、方块、梅花，小☺、大☻分别表示小王和大王。斗地主游戏共有三位玩家参与，首先将这54张牌的顺序打乱每人轮流摸一次牌，剩余3张留作底牌，然后在控制台打印三位玩家的牌和三张底牌。

**2.运行结果**

任务运行结果如图6-1所示：

手机屏幕截图

描述已自动生成

图6-1 运行结果图

### 【任务介绍】

* 学会分析“斗地主之洗牌发牌”任务的实现思路。
* 根据思路独立完成“斗地主之洗牌发牌”任务的源代码编写、编译及运行。
* 掌握List集合和Map集合特点及常用方法的使用。
* 掌握集合遍历的方式。

### 【实现思路】

（1）要实现纸牌程序，首先需要完成纸牌的组装。牌面是由花色（包括♠、♥、♦、♣花色）和数字（包括J、Q、K、A字母）两部分组成，可以创建两个ArrayList集合作为花色集合与数字集合，存储时需要注意。比10大的牌的数字用J、Q、K表示，1用A表示。

（2）将花色集合与数字集合这两个循环进行嵌套循环，将花色与数字组合，形成52章牌，并赋予其编号。将组合后的牌存放到一个HashMap集合中，集合的Key值是编号，value值是组装完成的纸牌。还有两张牌是大小王（小☺表示小王、大☻表示大王）。由于组装规则不一致，需单独使用add()方法将这两张牌加入到HashMap集合中。

（3）创建一个数字集合，用这个数字集合代替纸牌完成洗牌和发牌操作。由于纸牌的数量是54张，所以创建集合范围是0~53。

（4）可以使用Collection类的shuffle()方法完成打乱数字集合的操作，实现洗牌效果。由于只有3个人，所以可以使用for循环，通过将数字与3取余的方法，将代表不同纸牌的数字分配给不同人与底牌，实现发牌效果。

（5） 洗牌和发牌结束后，可以通过Collection类的sort()方法完成排序，之后通过foreach()循环HashMap集合，根据数字查找对应的纸牌字符串，并存入创建的字符串集合中，最后展示字符串集合。

### 【实现代码】

程序实现代码如文件6-1所示。

文件6-1 example5.java

1. package com.itheima;
2. import java.util.ArrayList;
3. import java.util.Collections;
4. import java.util.HashMap;
5. public class example5 {
6. public static void main(String[] args) {
7. // 准备花色
8. ArrayList<String> color = new ArrayList<String>();
9. color.add("♠");
10. color.add("♥");
11. color.add("♦");
12. color.add("♣");
13. // 准备数字 ,用ArrayList将纸牌由小到大排序
14. ArrayList<String> number = new ArrayList<String>();
15. for (int i = 3; i <= 10; i++) {
16. number.add(i + "");
17. }
18. number.add("J");
19. number.add("Q");
20. number.add("K");
21. number.add("A");
22. number.add("2");
23. // 定义一个map集合：用来将数字与每一张牌进行对应
24. HashMap<Integer, String> map = new HashMap<Integer, String>();
25. int index = 0;// 纸牌编号
26. for (String thisNumber : number) {// 循环纸牌数字
27. for (String thisColor : color) {// 循环纸牌花色
28. // 将花色与数字组合，形成52张牌，并赋予其编号
29. map.put(index++, thisColor + thisNumber);
30. }
31. }
32. // 加入大小王
33. map.put(index++, "小☺");
34. map.put(index++, "大☻");
35. // 一副54张的牌 ArrayList里边为0-53的数的新牌
36. ArrayList<Integer> cards = new ArrayList<Integer>();
37. for (int i = 0; i <= 53; i++) {
38. cards.add(i);// 此时的cards顺序为0-53
39. }
40. // 洗牌，使用Collections工具类中的shuffle()方法
41. Collections.shuffle(cards);// 此时的cards顺序已被打乱
42. // 创建三个玩家和底牌
43. ArrayList<Integer> iPlayer = new ArrayList<Integer>();
44. ArrayList<Integer> iPlayer2 = new ArrayList<Integer>();
45. ArrayList<Integer> iPlayer3 = new ArrayList<Integer>();
46. ArrayList<Integer> iSecretCards = new ArrayList<Integer>();
47. // 遍历这副洗好的牌，遍历过程中，将牌发到三个玩家和底牌中
48. for (int i = 0; i < cards.size(); i++) {
49. if (i >= 51) {
50. iSecretCards.add(cards.get(i));// 留取3张底牌
51. } else {
52. if (i % 3 == 0) {
53. iPlayer.add(cards.get(i));//与3取余为0的牌发给玩家1
54. } else if (i % 3 == 1) {
55. iPlayer2.add(cards.get(i));//与3取余为1的牌发给玩家2
56. } else {
57. iPlayer3.add(cards.get(i));// 其余的牌发给玩家3
58. }
59. }
60. }
61. // 对每个人手中的牌排序，使用Collections工具类中的sort()方法
62. Collections.sort(iPlayer);
63. Collections.sort(iPlayer2);
64. Collections.sort(iPlayer3);
65. // 对应数字形式的每个人手中的牌，定义字符串形式的牌
66. ArrayList<String> sPlayer = new ArrayList<String>();
67. ArrayList<String> sPlayer2 = new ArrayList<String>();
68. ArrayList<String> sPlayer3 = new ArrayList<String>();
69. ArrayList<String> sSecretCards = new ArrayList<String>();
70. // 循环主键，从map中获取纸牌
71. for (Integer key : iPlayer) {
72. sPlayer.add(map.get(key));
73. }
74. for (Integer key : iPlayer2) {
75. sPlayer2.add(map.get(key));
76. }
77. for (Integer key : iPlayer3) {
78. sPlayer3.add(map.get(key));
79. }
80. for (Integer key : iSecretCards) {
81. sSecretCards.add(map.get(key));
82. }
83. // 看牌
84. System.out.println("玩家1：" + sPlayer);
85. System.out.println("玩家2：" + sPlayer2);
86. System.out.println("玩家3：" + sPlayer3);
87. System.out.println("底牌：" + sSecretCards);
88. }
89. }

在文件6-1中，第8~22行代码将4种花色和13个纸牌编号分别放到两个ArrayList集合中。第24~34当代码foreach循环拼出52张纸牌，并将其依照从小到大的顺序放入Map集合中，并将大小王添加到其中。第36~41行代码创建了代替纸牌的数字集合，将其与纸牌中的序号对应，并将数字集合中的顺序打乱，此时集合相当于纸牌编号集合。第43~60行代码完成了将纸牌编号发给3位玩家并留3张纸牌编号做为底牌的操作。第62~64行代码将3位玩家手中纸牌编号进行排序。第66~88行代码通过循环3位玩家手中的纸牌编号，将纸牌字符串从Map中取出放入字符串集合。最后打印字符串集合，分别得到3个玩家的纸牌和底牌。