1. 写一个彩票的模拟程序：30选7。随机（1~30之间）生成7个随机数，注意不能重复。然后从键盘输入7个数，对比7个数是否与随机数有相同的。最后显示“中了几个号”。同时，如果中了7个号，显示一等奖；如果中了6个号，显示二等奖；如果中了5个号，显示三等奖。要求：首先在注释中写出程序的实现思想，特别是程序所使用的数据结构，然后写出Java实现代码。

核心代码：

|  |
| --- |
| **package** 实验四;  **import** java.util.\*;  **public** **class** LotteryTicket  {  **static** Scanner *sc* = **new** Scanner(System.***in***);  **public** **static** **void** main(String args[])  {  /\*HashSet 保证了元素的不重复性\*/  HashSet<Integer> ran = **new** HashSet<Integer>();  HashSet<Integer> res = **new** HashSet<Integer>();  /\* (1)先随机产生7个随机数，  \*(2)在通过键盘输入7个数  \*/  Random r=**new** Random();  **for** (**int** i = 0; i < 7; i++) ran.add(r.nextInt(30)+1);  **for** (**int** i = 1; i <= 7; i++) res.add(*sc*.nextInt());  /\*(3)两个集合取交集，最后result中剩下的就是最后中奖的号码  \*(4)集合的长度就是中奖的个数，最为几等奖的判定标准  \*/  HashSet<Integer> result = **new** HashSet<Integer>();  result.addAll(ran);  result.retainAll(res);//取交    System.***out***.println("中了" + result.size() + "个号");  **if** (result.size() == 7) System.***out***.println("一等奖");  **if** (result.size() == 6) System.***out***.println("二等奖");  **if** (result.size() == 5) System.***out***.println("三等奖");  }  } |

1. 用HashMap模拟一个网上购物车。要求：从键盘输入n本书的名称、单价、购买数量，将这些信息存入一个HashMap，然后将该HashMap作为参数调用方法getSum（HashMap books），该方法用于计算书的总价并返回。【说明：键盘输入可使用Scanner类】

|  |
| --- |
| **class** Book  {  **public** String name;  **public** **double** price;  **public** **int** number;  Book(String name, **double** price, **int** number)  {  **this**.name = name;  **this**.price = price;  **this**.number = number;  }  }  **public** **class** OnlineShopping  {  **static** Scanner *sc* = **new** Scanner(System.***in***);    **static** **double** getNum(HashMap<Integer, Book> books)  {  **double** total = 0;  **for** (**int** i = 1; i <= books.size(); i++)  {  Book b = books.get(i);  total += b.price \* (**double**)b.number;  }  **return** total;  }  **public** **static** **void** main(String args[])  {  HashMap<Integer, Book> mp = **new** HashMap<Integer, Book>();  **int** n = *sc*.nextInt();    **for** (**int** i = 1; i <= n; i++)  {  String name = *sc*.next();  **double** price = *sc*.nextDouble();  **int** number = *sc*.nextInt();  mp.put(i, **new** Book(name, price, number));  }    System.***out***.println("书的总价为：" + *getNum*(mp));  }  } |

1. 模拟保皇游戏开始的发牌过程。**规则**：4副扑克，5个玩家。1）有一个大王标记为皇上。每次发牌时，所发牌中有该大王的玩家是皇上。2）皇帝选择侍卫（也叫保儿、腿子，游戏过程中与皇帝一伙）：作为皇上的玩家从自己的牌中选择一张拥有相同三张（点数和花色都相同）的一张牌（不能是2、3、大小王），其他四个玩家中有这张牌的就是侍卫。例如，皇上有三个红桃5，其他四个玩家中有红桃5的玩家就是侍卫。特殊情况是：1）皇上有一套四张牌相同的点数的牌，皇帝可以自己做侍卫；2）皇帝没有满足要求的牌，无法获得侍卫。**程序要求**：程序启动后生成5个玩家，并自动给他们发牌。然后输出：1）皇帝和侍卫的名字及其手里的牌（每张牌输出为“花色”+“点数”，如红桃5，牌之间用“，”分割），并按照大王、小王、2、A、K、Q、J、10、9、8、7、6、5、4、3的顺序排列，相同点数但不同花色的牌要把相同花色的牌放在一起；2）那张作为侍卫所特有的牌（“花色”+“点数”）。如果无法得到侍卫，则程序输出相应的提示。例如，程序运行后输出如下的结果：

皇帝是：玩家1

皇帝的牌是：[皇上, 小王, 小王, 小王, 小王, 方片2, 黑桃2, 黑桃A, 黑桃A, 红桃A, 方片K, 梅花K, 黑桃K, 红桃K, 梅花Q, 梅花Q, 黑桃Q, **方片J, 方片J, 方片J**, 红桃J, 梅花9, 黑桃9, 黑桃9, 方片8, 梅花8, 红桃8, 梅花7, 黑桃7, 黑桃7, 红桃7, 梅花6, 梅花6, 黑桃6, 黑桃6, 方片5, 梅花5, 黑桃5, 黑桃5, 梅花4, 梅花4, 梅花4, 方片3, 红桃3]

侍卫对应的牌是：**方片J**

侍卫是：玩家2

侍卫的牌是：[方片2, 黑桃2, 红桃2, 方片A, 方片K, 梅花K, 梅花K, 黑桃K, 红桃K, 红桃K, 黑桃Q, 红桃Q, **方片J**, 方片10, 黑桃10, 红桃10, 红桃10, 红桃10, 方片9, 红桃9, 方片8, 梅花8, 梅花8, 黑桃8, 黑桃8, 黑桃8, 红桃8, 红桃8, 方片7, 黑桃7, 黑桃7, 方片6, 黑桃6, 黑桃5, 梅花4, 黑桃4, 红桃4, 红桃4, 方片3, 梅花3, 黑桃3, 红桃3, 红桃3]

核心代码：

|  |
| --- |
| **import** java.util.\*;  **public** **class** BaoHuang{  LinkedList<Card> cards = **null**;  LinkedList<Player> players = **null**;  **public** **static** **void** main(String[] args)  {  BaoHuang bh = **new** BaoHuang();  bh.send();  bh.check();  }  BaoHuang()  {  cards = **new** LinkedList<Card>();  players = **new** LinkedList<Player>();  **for** (**int** i = 0; i < 13; i++)  {  **for** (**int** j = 0; j < 4; j++)  {  **for** (**int** k = 0; k < 4; k++)  cards.add(**new** Card(j, i));// 基本牌  }  }  **for** (**int** i = 0; i < 4; i++)  {  cards.add(**new** Card(4, 14));// 小王  }  **for** (**int** i = 0; i < 3; i++)  {  cards.add(**new** Card(5, 14));// 大王  }  cards.add(**new** Card(6, 14));// 皇帝  **for** (**int** i = 0; i < 5; i++)  {  players.add(**new** Player());  }  }  **void** send()  {  Collections.*shuffle*(cards);  **int** csize = cards.size();  **for** (**int** i = 0; i < csize; i++)  {  sort(i);  }  }  **void** sort(**int** i)  {  **int** j = i % 5;  Player py = players.get(j);  LinkedList<Card> ll = py.pc;  Card cd = cards.get(i);  **if** (cd.type <= 3 && cd.value!=0 && cd.value!=12)  py.count[cd.type][cd.value]++;  **int** size = ll.size();  **boolean** bool = **false**;  **for** (i = 0; i < size; i++)  {  **if** (cd.compareTo(ll.get(i)) >= 0)  {  ll.add(i, cd);  bool = **true**;  **break**;  }  }  **if** (!bool)  {  ll.add(cd);  }  }  **void** check()  {    Player temp = **null**;    **for** (**int** i = 0; i < 5; i++)  {  temp = players.get(i);  **if** ("皇帝".equals(temp.pc.get(0).toString()))  {  System.***out***.println("皇帝是：玩家"+(i+1));  System.***out***.print("皇帝的牌是：");  temp.show();  **break**;  }  }  **boolean** bool = **false**;  **for** (**int** i = 0; i < 4; i++)  {  **for** (**int** j = 1; j < 13; j++)  {  **if** (temp.count[i][j] >= 3)  {  bool = **true**;  **for**(**int** k=0; k<5; k++)  {  **if**(players.get(k).count[i][j]==1)  {  System.***out***.println("侍卫对应的牌是："+Card.*change*(i,j));  System.***out***.println("侍卫是：玩家"+(k+1));  System.***out***.print("侍卫的牌是：");  players.get(k).show();  **break**;  }  }  }  **if**(bool)**break**;  }  **if**(bool)**break**;  }  **if**(!bool)System.***out***.println("没有侍卫");  }  }  **class** Card **implements** Comparable<Card>{    **int** type = 0;  **int** value = 0;    **public** Card(**int** type, **int** value)  {  **this**.type = type;  **this**.value = value;  }    **static** String change(**int** type,**int** value)  {  String s1 = **null**;  String s2 = **null**;  **switch** (value)  {  **case** 8:  s2 = "J";  **break**;  **case** 9:  s2 = "Q";  **break**;  **case** 10:  s2 = "K";  **break**;  **case** 11:  s2 = "A";  **break**;  **case** 12:  s2 = "2";  **break**;  **default**:  s2 = (value + 3) + "";  }  **switch** (type)  {  **case** 2:  s1 = "梅花 ";  **break**;  **case** 3:  s1 = "方片 ";  **break**;  **case** 0:  s1 = "红桃 ";  **break**;  **case** 1:  s1 = "黑桃 ";  **break**;  **case** 4:  s1 = "小王";  s2 = "";  **break**;  **case** 5:  s1 = "大王";  s2 = "";  **break**;  **case** 6:  s1 = "皇帝";  s2 = "";  **break**;  **default**:  }  **return** s1 + s2;  }  @Override  **public** String toString()  {  **return** *change*(type,value);  }    @Override  **public** **int** compareTo(Card obj)  {  **if** (obj == **this**)  **return** 0;  Card other = (Card) obj;  **if** (**this**.value < other.value)  {  **return** -1;  }  **else** **if** (**this**.value > other.value)  {  **return** 1;  }  **else** **if** (**this**.type < other.type)  {  **return** -1;  }  **else** **if** (**this**.type > other.type)  {  **return** 1;  }  **else**  {  **return** 0;  }  }  @Override  **public** **int** hashCode()  {  **final** **int** prime = 52;  **int** result = 1;  result = prime \* type;  result = result + value;  **return** result;  }    @Override  **public** **boolean** equals(Object obj)  {  **if** (**this** == obj)  **return** **true**;  **if** (obj == **null**)  **return** **false**;  **if** (getClass() != obj.getClass())  **return** **false**;  Card other = (Card) obj;  **if** (type != other.type)  **return** **false**;  **if** (value != other.value)  **return** **false**;  **return** **true**;  }    }  **class** Player{  LinkedList<Card> pc = **new** LinkedList<Card>();  **int**[][] count = **new** **int**[4][13];  Player()  {  **for** (**int** i = 0; i < 4; i++)  {  **for** (**int** j = 0; j < 13; j++)  {  count[i][j] = 0;  }  }  }  **void** show()  {  System.***out***.print("["+pc.get(0));  **for** (**int** i = 1; i < pc.size(); i++)  {  System.***out***.print("," + pc.get(i));  }  System.***out***.println("]");  }  } |