JAVA语言程序设计A/B，A卷参考答案

（2017年上学期）

（**121151-2，123151-2，191151-5**）

一、名词解释**（每题2分，共5小题，总分10分）**

1.继承：可以基于已经存在的类构造一个新类。继承已经存在的类就可以复用这些类的方法和域。在此基础上，可以添加新的方法和域，从而扩充了类的功能。

2.向量(Vector)是java.util类包提供的一个工具类。它是允许不同类型元素共存的变长数组,Vector实现了变长数组。

3. 类方法，也称静态方法，指的是用static关键字修饰的方法。此方法属类本身的方法，不属于类的某一个实例（对象）。类方法中不可直接使用实例变量。其调用方式有三种：可直接调用、类名.方法名、对象名.方法名。。

4. DNS（Domain Name System，域名系统），因特网上作为域名和IP地址相互映射的一个分布式数据库，能够使用户更方便的访问互联网，而不用去记住能够被机器直接读取的IP数串。通过主机名，最终得到该主机名对应的IP地址的过程叫做域名解析（或主机名解析）。。

5. 异常是程序运行过程中出现的错误。Exception是程序本身可以处理的异常，这种异常分两大类运行时异常和非运行时异常。程序中应当尽可能去处理这些异常。

二、单选**（每题1分，共20小题，总分20分）**

01-05： ADAAD 06-10：CBCDD

11-15：CDCAA 16-20： DAACC

三、多选**（每题2分，共10小题，总分20分）**

01 BCD 02ACD 03 ABCD 04 AD 05 ABCD

1. ABC 07 BC 08 ABC 09 BCD 10 BD
2. 写程序的运行结果**（每题4分，共5小题，总分20分）**

1. 11

2. BADCADCA

3.从键盘输入10个数，按大到小排序

4.119

5. 2017

五、编程题（略）**（共3大题，每题10分，总分30分）**

1.

1. **void** SegmentPrime(**int** L, **int** U)
2. {
3. // 求区间[L, U]中的素数.
4. **int** i, j;
5. **int** SU = sqrt(1.0 \* U);
6. **int** d = U - L + 1;
7. **for** (i = 0; i < d; i++) IsNotPrime[i] = 0; // 一开始全是素数.
8. **for** (i = (L % 2 != 0); i < d; i += 2) IsNotPrime[i] = 1; // 把偶数的直接去掉.
9. **for** (i = 3; i <= SU; i += 2)
10. {
11. **if** (i > L && IsNotPrime[i - L]) **continue**; // IsNotPrime[i - L] == 1说明i不是素数.
12. j = (L / i) \* i; // j为i的倍数，且最接近L的数.
13. **if** (j < L) j += i;
14. **if** (j == i) j += i; // i为素数，j = i说明j也是素数，所以直接 + i.
15. j = j - L;
16. **for** (; j < d; j += i) IsNotPrime[j] = 1; // 说明j不是素数(IsNotPrime[j - L] = 1).
17. }
18. **if** (L <= 1) IsNotPrime[1 - L] = 1;
19. **if** (L <= 2) IsNotPrime[2 - L] = 0;
20. PrimeNum = 0;
21. **for** (i = 0; i < d; i++) **if** (!IsNotPrime[i]) PrimeList[PrimeNum++] = i + L;
22. }

2.

**interface** ordinary //普通用户

{

**void** dining(); //普通餐厅

}

**interface** vip //vip用户

{

**void** vipdining();//高级餐厅

}

**interface** trial **extends** ordinary,vip

{

**void** dining();

**void** vipdining();

}

**class** user **implements** ordinary

{

**public** **void** dining()

{

System.*out*.println("我是普通用户,我只能去普通餐厅消费!");

}

}

**class** vipuser **implements** vip

{

**public** **void** vipdining()

{

System.*out*.println("我是vip用户,我只能在高级餐厅消费!");

}

}

**class** trialuser **implements** trial

{

**public** **void** dining()

{

System.*out*.println("我是试用用户,我可以进普通餐厅消费!");

};

**public** **void** vipdining()

{

System.*out*.println("我是试用用户,我可以进高通餐厅消费!");

};

}

**public** **class** ExtendsInterface {

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

{

ordinary o = **new** user();

o.dining();

vip v = **new** vipuser();

v.vipdining();

trial t = **new** trialuser();

t.dining();

t.vipdining();

}

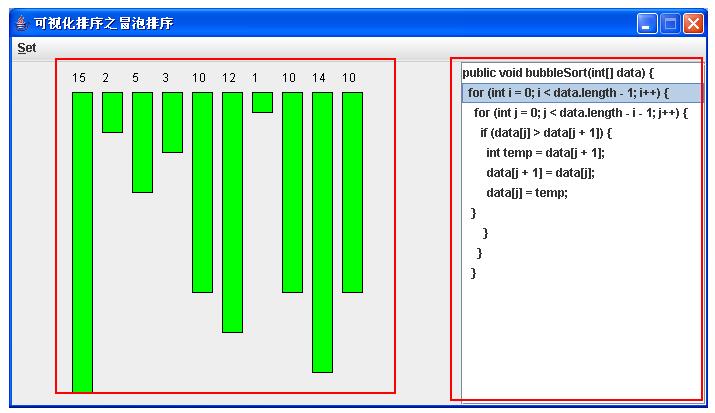
}

3.回文串

1. **public** **class** Main {
3. **public** **static** **void** main(String[] args) {
5. Scanner input = **new** Scanner(System.in);
6. String str = input.nextLine();
7. System.out.println(isHuiWen(str));
8. }
10. **public** **static** **boolean** isHuiWen(String s) {
11. StringBuffer sb = **new** StringBuffer();
12. **for** (**int** i = 0; i < s.length(); i++) {
13. **if** (Character.isLetter(s.charAt(i))
14. || Character.isDigit(s.charAt(i))) {
15. sb.append(s.charAt(i));
17. }
18. }
19. String str2 = sb.toString().toLowerCase();
20. System.out.println(str2);
21. sb.delete(0, sb.length());
22. **for** (**int** j = 0; j <= (str2.length() + 1) / 2 - 1; j++) {
23. sb.append(str2.charAt(j) == str2.charAt(str2.length() - 1 - j));
25. }// for
26. **return** !sb.toString().contains("f");
28. }// /isHuiWen
30. }

计算器

1. **public** **class** Calculator **extends** JFrame **implements** ActionListener {
2. /\*\* 计算器上的键的显示名字 \*/
3. **private** **final** String[] KEYS = { "7", "8", "9", "/", "sqrt", "4", "5", "6",
4. "\*", "%", "1", "2", "3", "-", "1/x", "0", "+/-", ".", "+", "=" };
5. /\*\* 计算器上的功能键的显示名字 \*/
6. **private** **final** String[] COMMAND = { "Backspace", "CE", "C" };
7. /\*\* 计算器左边的M的显示名字 \*/
8. **private** **final** String[] M = { " ", "MC", "MR", "MS", "M+" };
9. /\*\* 计算器上键的按钮 \*/
10. **private** JButton keys[] = **new** JButton[KEYS.length];
11. /\*\* 计算器上的功能键的按钮 \*/
12. **private** JButton commands[] = **new** JButton[COMMAND.length];
13. /\*\* 计算器左边的M的按钮 \*/
14. **private** JButton m[] = **new** JButton[M.length];
15. /\*\* 计算结果文本框 \*/
16. **private** JTextField resultText = **new** JTextField("0");
18. // 标志用户按的是否是整个表达式的第一个数字,或者是运算符后的第一个数字
19. **private** **boolean** firstDigit = **true**;
20. // 计算的中间结果。
21. **private** **double** resultNum = 0.0;
22. // 当前运算的运算符
23. **private** String operator = "=";
24. // 操作是否合法
25. **private** **boolean** operateValidFlag = **true**;
27. /\*\*
28. \* 构造函数
29. \*/
30. **public** Calculator() {
31. **super**();
32. // 初始化计算器
33. init();
34. // 设置计算器的背景颜色
35. **this**.setBackground(Color.LIGHT\_GRAY);
36. **this**.setTitle("计算器");
37. // 在屏幕(500, 300)坐标处显示计算器
38. **this**.setLocation(500, 300);
39. // 不许修改计算器的大小
40. **this**.setResizable(**false**);
41. // 使计算器中各组件大小合适
42. **this**.pack();
43. }
45. /\*\*
46. \* 初始化计算器
47. \*/
48. **private** **void** init() {
49. // 文本框中的内容采用右对齐方式
50. resultText.setHorizontalAlignment(JTextField.RIGHT);
51. // 不允许修改结果文本框
52. resultText.setEditable(**false**);
53. // 设置文本框背景颜色为白色
54. resultText.setBackground(Color.white);
56. // 初始化计算器上键的按钮，将键放在一个画板内
57. JPanel calckeysPanel = **new** JPanel();
58. // 用网格布局器，4行，5列的网格，网格之间的水平方向间隔为3个象素，垂直方向间隔为3个象素
59. calckeysPanel.setLayout(**new** GridLayout(4, 5, 3, 3));
60. **for** (**int** i = 0; i < KEYS.length; i++) {
61. keys[i] = **new** JButton(KEYS[i]);
62. calckeysPanel.add(keys[i]);
63. keys[i].setForeground(Color.blue);
64. }
65. // 运算符键用红色标示，其他键用蓝色表示
66. keys[3].setForeground(Color.red);
67. keys[8].setForeground(Color.red);
68. keys[13].setForeground(Color.red);
69. keys[18].setForeground(Color.red);
70. keys[19].setForeground(Color.red);
72. // 初始化功能键，都用红色标示。将功能键放在一个画板内
73. JPanel commandsPanel = **new** JPanel();
74. // 用网格布局器，1行，3列的网格，网格之间的水平方向间隔为3个象素，垂直方向间隔为3个象素
75. commandsPanel.setLayout(**new** GridLayout(1, 3, 3, 3));
76. **for** (**int** i = 0; i < COMMAND.length; i++) {
77. commands[i] = **new** JButton(COMMAND[i]);
78. commandsPanel.add(commands[i]);
79. commands[i].setForeground(Color.red);
80. }
82. // 初始化M键，用红色标示，将M键放在一个画板内
83. JPanel calmsPanel = **new** JPanel();
84. // 用网格布局管理器，5行，1列的网格，网格之间的水平方向间隔为3个象素，垂直方向间隔为3个象素
85. calmsPanel.setLayout(**new** GridLayout(5, 1, 3, 3));
86. **for** (**int** i = 0; i < M.length; i++) {
87. m[i] = **new** JButton(M[i]);
88. calmsPanel.add(m[i]);
89. m[i].setForeground(Color.red);
90. }
92. // 下面进行计算器的整体布局，将calckeys和command画板放在计算器的中部，
93. // 将文本框放在北部，将calms画板放在计算器的西部。
95. // 新建一个大的画板，将上面建立的command和calckeys画板放在该画板内
96. JPanel panel1 = **new** JPanel();
97. // 画板采用边界布局管理器，画板里组件之间的水平和垂直方向上间隔都为3象素
98. panel1.setLayout(**new** BorderLayout(3, 3));
99. panel1.add("North", commandsPanel);
100. panel1.add("West", calckeysPanel);
102. // 建立一个画板放文本框
103. JPanel top = **new** JPanel();
104. top.setLayout(**new** BorderLayout());
105. top.add("Center", resultText);
107. // 整体布局
108. getContentPane().setLayout(**new** BorderLayout(3, 5));
109. getContentPane().add("North", top);
110. getContentPane().add("Center", panel1);
111. getContentPane().add("West", calmsPanel);
112. // 为各按钮添加事件侦听器
113. // 都使用同一个事件侦听器，即本对象。本类的声明中有implements ActionListener
114. **for** (**int** i = 0; i < KEYS.length; i++) {
115. keys[i].addActionListener(**this**);
116. }
117. **for** (**int** i = 0; i < COMMAND.length; i++) {
118. commands[i].addActionListener(**this**);
119. }
120. **for** (**int** i = 0; i < M.length; i++) {
121. m[i].addActionListener(**this**);
122. }
123. }
125. /\*\*
126. \* 处理事件
127. \*/
128. **public** **void** actionPerformed(ActionEvent e) {
129. // 获取事件源的标签
130. String label = e.getActionCommand();
131. **if** (label.equals(COMMAND[0])) {
132. // 用户按了"Backspace"键
133. handleBackspace();
134. } **else** **if** (label.equals(COMMAND[1])) {
135. // 用户按了"CE"键
136. resultText.setText("0");
137. } **else** **if** (label.equals(COMMAND[2])) {
138. // 用户按了"C"键
139. handleC();
140. } **else** **if** ("0123456789.".indexOf(label) >= 0) {
141. // 用户按了数字键或者小数点键
142. handleNumber(label);
143. // handlezero(zero);
144. } **else** {
145. // 用户按了运算符键
146. handleOperator(label);
147. }
148. }
150. /\*\*
151. \* 处理Backspace键被按下的事件
152. \*/
153. **private** **void** handleBackspace() {
154. String text = resultText.getText();
155. **int** i = text.length();
156. **if** (i > 0) {
157. // 退格，将文本最后一个字符去掉
158. text = text.substring(0, i - 1);
159. **if** (text.length() == 0) {
160. // 如果文本没有了内容，则初始化计算器的各种值
161. resultText.setText("0");
162. firstDigit = **true**;
163. operator = "=";
164. } **else** {
165. // 显示新的文本
166. resultText.setText(text);
167. }
168. }
169. }
171. /\*\*
172. \* 处理数字键被按下的事件
173. \*
174. \* @param key
175. \*/
176. **private** **void** handleNumber(String key) {
177. **if** (firstDigit) {
178. // 输入的第一个数字
179. resultText.setText(key);
180. } **else** **if** ((key.equals(".")) && (resultText.getText().indexOf(".") < 0)) {
181. // 输入的是小数点，并且之前没有小数点，则将小数点附在结果文本框的后面
182. resultText.setText(resultText.getText() + ".");
183. } **else** **if** (!key.equals(".")) {
184. // 如果输入的不是小数点，则将数字附在结果文本框的后面
185. resultText.setText(resultText.getText() + key);
186. }
187. // 以后输入的肯定不是第一个数字了
188. firstDigit = **false**;
189. }
191. /\*\*
192. \* 处理C键被按下的事件
193. \*/
194. **private** **void** handleC() {
195. // 初始化计算器的各种值
196. resultText.setText("0");
197. firstDigit = **true**;
198. operator = "=";
199. }
201. /\*\*
202. \* 处理运算符键被按下的事件
203. \*
204. \* @param key
205. \*/
206. **private** **void** handleOperator(String key) {
207. **if** (operator.equals("/")) {
208. // 除法运算
209. // 如果当前结果文本框中的值等于0
210. **if** (getNumberFromText() == 0.0) {
211. // 操作不合法
212. operateValidFlag = **false**;
213. resultText.setText("除数不能为零");
214. } **else** {
215. resultNum /= getNumberFromText();
216. }
217. } **else** **if** (operator.equals("1/x")) {
218. // 倒数运算
219. **if** (resultNum == 0.0) {
220. // 操作不合法
221. operateValidFlag = **false**;
222. resultText.setText("零没有倒数");
223. } **else** {
224. resultNum = 1 / resultNum;
225. }
226. } **else** **if** (operator.equals("+")) {
227. // 加法运算
228. resultNum += getNumberFromText();
229. } **else** **if** (operator.equals("-")) {
230. // 减法运算
231. resultNum -= getNumberFromText();
232. } **else** **if** (operator.equals("\*")) {
233. // 乘法运算
234. resultNum \*= getNumberFromText();
235. } **else** **if** (operator.equals("sqrt")) {
236. // 平方根运算
237. resultNum = Math.sqrt(resultNum);
238. } **else** **if** (operator.equals("%")) {
239. // 百分号运算，除以100
240. resultNum = resultNum / 100;
241. } **else** **if** (operator.equals("+/-")) {
242. // 正数负数运算
243. resultNum = resultNum \* (-1);
244. } **else** **if** (operator.equals("=")) {
245. // 赋值运算
246. resultNum = getNumberFromText();
247. }
248. **if** (operateValidFlag) {
249. // 双精度浮点数的运算
250. **long** t1;
251. **double** t2;
252. t1 = (**long**) resultNum;
253. t2 = resultNum - t1;
254. **if** (t2 == 0) {
255. resultText.setText(String.valueOf(t1));
256. } **else** {
257. resultText.setText(String.valueOf(resultNum));
258. }
259. }
260. // 运算符等于用户按的按钮
261. operator = key;
262. firstDigit = **true**;
263. operateValidFlag = **true**;
264. }
266. /\*\*
267. \* 从结果文本框中获取数字
268. \*
269. \* @return
270. \*/
271. **private** **double** getNumberFromText() {
272. **double** result = 0;
273. **try** {
274. result = Double.valueOf(resultText.getText()).doubleValue();
275. } **catch** (NumberFormatException e) {
276. }
277. **return** result;
278. }
280. **public** **static** **void** main(String args[]) {
281. Calculator calculator1 = **new** Calculator();
282. calculator1.setVisible(**true**);
283. calculator1.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
284. }
285. }

冒泡排序可视化  
本文简单地实现了冒泡排序的可视化。界面很简单就包括两个部分：界面左侧是可视化排序部分，右侧是冒泡排序的代码。如下图所示：   
  
  
  
程序的**关键点**主要有两点：   
1. 如何在页面上表示出排序程序的运行过程。   
2. 如何将排序程序的运行过程和可视化排序结合起来，保持状态一致。   
  
**我的解决方法如下：**   
我采用了**JList**去模拟程序的执行，JList有一个setSelectedIndex的方法，能高亮显示指定的行。通过改变selectedIndex的值，能够达到模拟程序执行的效果。在这个过程中，记录下两个循环的索引状态值，根据这些状态值去调整可视化排序。

1. **public** **class** BubbleSortVisualizationFrame **extends** JFrame {
3. **private** **static** **final** **long** serialVersionUID = -6725108659717827278L;
5. **private** Container contentPane;
7. /\*\*
8. \* 设置三个Menu Item，分别用于开始程序，调整运行的速度以及退出程序
9. \*
10. \*/
11. **private** JMenuItem startMI = **new** JMenuItem("Start");
13. **private** JMenu speedMenu = **new** JMenu("Speed");
15. **private** JMenuItem exitMI = **new** JMenuItem("Exit");
17. /\*\*
18. \* 设定5个速度级别
19. \*/
20. **private** JRadioButtonMenuItem speedMI1 = **new** JRadioButtonMenuItem("Speed1",
21. **true**);
23. **private** JRadioButtonMenuItem speedMI2 = **new** JRadioButtonMenuItem("Speed2",
24. **false**);
26. **private** JRadioButtonMenuItem speedMI3 = **new** JRadioButtonMenuItem("Speed3",
27. **false**);
29. **private** JRadioButtonMenuItem speedMI4 = **new** JRadioButtonMenuItem("Speed4",
30. **false**);
32. **private** JRadioButtonMenuItem speedMI5 = **new** JRadioButtonMenuItem("Speed5",
33. **false**);
35. **public** **int** speedFlag = 1;
37. /\*\*
38. \* 冒泡排序可视化的Panel
39. \*/
40. **private** BubbleSortPanel panel;
42. **public** BubbleSortVisualizationFrame(){
44. setTitle("可视化排序之冒泡排序");
45. setSize(700, 400);
46. setResizable(**false**);
48. JMenuBar menuBar = **new** JMenuBar();
49. setJMenuBar(menuBar);
51. JMenu setMenu = **new** JMenu("Set");
53. setMenu.setMnemonic('s');
55. menuBar.add(setMenu);
57. setMenu.add(startMI);
58. setMenu.addSeparator();
60. setMenu.addSeparator();
61. setMenu.add(speedMenu);
62. setMenu.addSeparator();
63. setMenu.add(exitMI);
65. ButtonGroup group = **new** ButtonGroup();
66. group.add(speedMI1);
67. group.add(speedMI2);
68. group.add(speedMI3);
69. group.add(speedMI4);
70. group.add(speedMI5);
72. speedMenu.add(speedMI1);
73. speedMenu.add(speedMI2);
74. speedMenu.add(speedMI3);
75. speedMenu.add(speedMI4);
76. speedMenu.add(speedMI5);
78. startMI.addActionListener(**new** StartAction());
79. speedMI1.addActionListener(**new** SpeedAction());
80. speedMI2.addActionListener(**new** SpeedAction());
81. speedMI3.addActionListener(**new** SpeedAction());
82. speedMI4.addActionListener(**new** SpeedAction());
83. speedMI5.addActionListener(**new** SpeedAction());
84. exitMI.addActionListener(**new** ExitAction());
86. contentPane = getContentPane();
88. panel = **new** BubbleSortPanel(**this**);
89. contentPane.add(panel);
90. startMI.setEnabled(**true**);
91. }
93. **private** **class** StartAction **implements** ActionListener {
94. **public** **void** actionPerformed(ActionEvent event) {
95. startMI.setEnabled(**false**);
96. panel.timer.start();
97. }
98. }
100. **private** **class** ExitAction **implements** ActionListener {
101. **public** **void** actionPerformed(ActionEvent event) {
102. System.exit(0);
103. }
104. }
106. **private** **class** SpeedAction **implements** ActionListener {
107. **public** **void** actionPerformed(ActionEvent event) {
108. Object speed = event.getSource();
109. **if** (speed == speedMI1) {
110. speedFlag = 1;
111. } **else** **if** (speed == speedMI2) {
112. speedFlag = 2;
113. } **else** **if** (speed == speedMI3) {
114. speedFlag = 3;
115. } **else** **if** (speed == speedMI4) {
116. speedFlag = 4;
117. } **else** **if** (speed == speedMI5) {
118. speedFlag = 5;
119. }
121. panel.timer.setDelay(1000 - 200 \* (speedFlag - 1));
122. }
123. }
125. }

**Java代码  [收藏代码](javascript:void())**

1. **package** my.visualization.sort.bubble;
2. **import** java.awt.BorderLayout;
3. **import** java.awt.Color;
4. **import** java.awt.Graphics;
5. **import** java.awt.Graphics2D;
6. **import** java.awt.event.ActionEvent;
7. **import** java.awt.event.ActionListener;
8. **import** java.io.Serializable;
9. **import** java.util.ArrayList;
10. **import** java.util.List;
11. **import** java.util.Random;
13. **import** javax.swing.JList;
14. **import** javax.swing.JPanel;
15. **import** javax.swing.JScrollPane;
16. **import** javax.swing.Timer;

19. **public** **class** BubbleSortPanel **extends** JPanel{
21. **private** **static** **final** **long** serialVersionUID = -9149581857139587792L;
23. **private** **static** **final** String[] code = {
24. "public void bubbleSort(int[] data) {       ",
25. "  for (int i = 0; i < data.length - 1; i++) {",
26. "    for (int j = 0; j < data.length - i - 1; j++) {",
27. "      if (data[j] > data[j + 1]) {          ",
28. "        int temp = data[j + 1];         ",
29. "        data[j + 1] = data[j];           ",
30. "        data[j] = temp;             ",
31. "      }                              ",
32. "       }                                ",
33. "     }                                  ",
34. "   }                                    "
35. };
37. /\*\*
38. \* 初始化10个数据
39. \*/
40. **private** List<NumberRectangle> numbers = initialNumberRectangles();
42. **private** JList codeList = **new** JList(code);;
44. **public** TimerAction timerAction;
46. **public** Timer timer;
48. **public** BubbleSortVisualizationFrame frame;
50. **public** BubbleSortPanel(BubbleSortVisualizationFrame frame) {
52. timerAction = **new** TimerAction();
53. timer = **new** Timer(1000, timerAction);
55. codeList.setSelectedIndex(1);
56. JScrollPane scrollPane1 = **new** JScrollPane(codeList);
57. **this**.setLayout(**new** BorderLayout());
58. **this**.add(scrollPane1, BorderLayout.EAST);
60. **this**.frame = frame;
61. }

64. /\*\*
65. \* 判断排序是否已经结束
66. \*/
67. **private** **boolean** completed = **false**;
69. **public** **void** paintComponent(Graphics g) {
70. **super**.paintComponent(g);
71. Graphics2D g2 = (Graphics2D) g;
73. drawNumberRectangles(g2);
74. }
76. **private** **void** drawNumberRectangles(Graphics2D g2){
77. **for**(NumberRectangle rectangle: numbers){
78. rectangle.draw(g2);
79. }
80. }
82. **int** outerLoop = 0;
83. **int** innerLoop = 0;
84. **int** selectedIndex = 1;

87. **private** **class** TimerAction **implements** ActionListener, Serializable {
89. **private** **static** **final** **long** serialVersionUID = -8671813189049345697L;
91. **public** **void** actionPerformed(ActionEvent event) {
92. **if** (completed) {
93. **return**;
94. }
96. **switch**(selectedIndex){
97. **case** 1:
98. **if**(outerLoop<10){
99. innerLoop = 0;
100. codeList.setSelectedIndex(selectedIndex++);
101. }**else**{
102. selectedIndex = 10;
103. }
104. **break**;
105. **case** 2:
106. **if**(innerLoop < 10 - outerLoop -1){
107. numbers.get(innerLoop).setColor(Color.RED);
108. numbers.get(innerLoop+1).setColor(Color.BLUE);
109. codeList.setSelectedIndex(selectedIndex++);
110. }**else**{
111. outerLoop++;
112. selectedIndex =1;
113. }
114. **break**;
115. **case** 3:
116. **if**(numbers.get(innerLoop).getValue()>numbers.get(innerLoop+1).getValue()){
117. codeList.setSelectedIndex(selectedIndex++);
118. }**else**{
119. numbers.get(innerLoop+1).setColor(Color.GREEN);
120. numbers.get(innerLoop).setColor(Color.GREEN);
121. innerLoop++;
122. selectedIndex = 2;
123. }
124. **break**;
125. **case** 4:
126. codeList.setSelectedIndex(selectedIndex++);
127. **break**;
128. **case** 5:
129. codeList.setSelectedIndex(selectedIndex++);
130. **break**;
132. **case** 6:
133. codeList.setSelectedIndex(selectedIndex);
134. **int** tempValue1 = numbers.get(innerLoop).getValue();
135. **int** tempValue2 = numbers.get(innerLoop+1).getValue();
136. numbers.get(innerLoop+1).setValue(tempValue1);
137. numbers.get(innerLoop).setValue(tempValue2);
138. numbers.get(innerLoop+1).setColor(Color.GREEN);
139. numbers.get(innerLoop).setColor(Color.GREEN);
140. selectedIndex = 2;
141. innerLoop++;
142. **break**;
144. **case** 10:
145. **if**(selectedIndex ==10){
146. completed = **true**;
147. codeList.setSelectedIndex(selectedIndex);
148. }
149. **break**;
150. **default**: **break**;
151. }
153. repaint();

156. }
157. }
159. **private** List<NumberRectangle> initialNumberRectangles(){
160. List<NumberRectangle> list = **new** ArrayList<NumberRectangle>();
161. /\*\*
162. \* 随机产生10个数组
163. \*/
164. Random random = **new** Random();
165. **for**(**int** i=1; i<=10;i++){
166. list.add(**new** NumberRectangle(i,1,random.nextInt(15)+1, Color.GREEN));
167. }
168. **return** list;
169. }


173. }

**Java代码  [收藏代码](javascript:void())**

1. **package** my.visualization.sort.bubble;
3. **import** java.awt.Color;
4. **import** java.awt.Graphics2D;
5. **import** java.awt.geom.Rectangle2D;
7. **public** **class** NumberRectangle {
9. **private** **int** x;
11. **private** **int** y;
13. **private** **int** value;
15. **private** Color color;
17. **public** NumberRectangle() {
18. }
20. **public** NumberRectangle(**int** x, **int** y, **int** value, Color color) {
21. **this**.x = x;
22. **this**.y = y;
23. **this**.color = color;
24. **this**.value = value;
26. }
28. **public** **void** draw(Graphics2D g2) {
29. **int** clientX = 30 + x \* 30;
30. **int** clientY = 20 + y \* 10;
31. Rectangle2D.Double rect = **new** Rectangle2D.Double(clientX, clientY, 20,
32. value \* 20);
33. g2.setPaint(color);
34. g2.fill(rect);
35. g2.setPaint(Color.BLACK);
36. g2.draw(rect);
37. g2.drawString(String.valueOf(value), clientX, clientY - 10);
38. }
40. /\*\*
41. \* @return the color
42. \*/
43. **public** Color getColor() {
44. **return** color;
45. }
47. /\*\*
48. \* @param color
49. \*            the color to set
50. \*/
51. **public** **void** setColor(Color color) {
52. **this**.color = color;
53. }
55. /\*\*
56. \* @return the x
57. \*/
58. **public** **int** getX() {
59. **return** x;
60. }
62. /\*\*
63. \* @param x
64. \*            the x to set
65. \*/
66. **public** **void** setX(**int** x) {
67. **this**.x = x;
68. }
70. /\*\*
71. \* @return the y
72. \*/
73. **public** **int** getY() {
74. **return** y;
75. }
77. /\*\*
78. \* @param y
79. \*            the y to set
80. \*/
81. **public** **void** setY(**int** y) {
82. **this**.y = y;
83. }
85. /\*\*
86. \* @return the value
87. \*/
88. **public** **int** getValue() {
89. **return** value;
90. }
92. /\*\*
93. \* @param value
94. \*            the value to set
95. \*/
96. **public** **void** setValue(**int** value) {
97. **this**.value = value;
98. }
100. }

**Java代码  [收藏代码](javascript:void())**

1. **package** my.visualization.sort.bubble;
3. **import** javax.swing.JFrame;
5. **public** **class** SortApplication {
6. @SuppressWarnings("deprecation")
7. **public** **static** **void** main(String[] args) {
8. BubbleSortVisualizationFrame frame = **new** BubbleSortVisualizationFrame();
9. frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
10. frame.show();
11. }
12. }