

목차

Order/OrderList 의 수정	2
OrderManagement 의 수정	3
OrderInfo 의 수정	7
디버깅	8
최종 실행 결과	10
평가표	15
소스 코드	16

1. Order/OrderList의 수정

1) Order: setAddress()/setItemStatus()

OrderManagement의 수정 모드에서 사용할 두 함수를 정의한다. 이 두 함수를 통해 각각 사용자가 특정 주소와 제품의 상태를 변경할 수 있게 한다.

```
79 public void setAddress(String addr) {
80     _location = addr;
81 }
82
83 public void setItemStatus(int i, String status) {
84     if (i < _numItem) _items[i]._status = status;
85 }
```

2) Order: set()의 수정

Set()의 fields[] 관련 조건문에서 성립을 요구하는 조건을 모두 (fields[n] == "" || fields[n].length() == 0)으로 수정한다. 이는 OrderInfo의 ADD에서 아무것도 입력하지 않고 저장할 때 null은 아니지만 길이가 0인 String을 받는 것을 피하기 위함이다.

```
87 private void set(String[] fields) throws Exception {
88     String[] tok = fields[4].split(":");
89     _numItem = 0;
90     for (int i = 0; i < tok.length; i++) {
91         _items[i] = new Item(tok[i]);
92         _numItem++;
93     }
94     LocalDateTime now = LocalDateTime.now();
95     if (fields[0] == "" || fields[0].length() == 0) {
96         _id = now.format(DateTimeFormatter.ofPattern("yyyyMMddHHmmss"));
97     } else {
98         _id = fields[0];
99     }
100     _name = fields[1];
101     if (fields[2] == "" || fields[2].length() == 0) {
102         _time = now.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm"));
103     } else {
104         SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm");
105         sdf.setLenient(false);
106         try {
107             Date date = sdf.parse(fields[2]);
108             _time = fields[2];
109         } catch (ParseException e) {
110             System.out.println("Date Format Error -- " + fields[2]);
111             throw new TimeException();
112         }
113     }
114     int price_sum = 0;
115     for (int i = 0; i < _numItem; i++) {
116         price_sum += _items[i]._price;
117     }
118     if (fields[3] == "" || fields[3].length() == 0) {
119         _price = price_sum;
120     } else {
121         _price = Integer.parseInt(fields[3]);
122         if (_price != price_sum) {
123             System.out.println("ERROR: price mismatched");
124         }
125     }
126     _location = fields[5];
127 }
```

3) OrderList: saveOrders()

try-catch를 통해 파일을 받아 수정한다. 파일이 존재하지 않으면 새로운 텍스트 파일을 생성하며, 파일이 존재하면 file에 입력받은 String을 출력하고 오류가 존재한다면 Exception을 전송한다.

```
79 //untested
80 public void saveOrders(String orderFileName) throws Exception{
81     try {
82         File file = new File(orderFileName);
83         if (!file.exists()) {
84             file.createNewFile();
85         }
86         print(new PrintStream(file));
87     } catch (Exception e) {
88         System.out.println(e.getMessage());
89         throw new Exception();
90     }
91 }
92 }
```

4) OrderList: sortByDate()/sortByCustomer()

날짜 또는 고객 이름을 통한 정렬은 for loop를 통한 bubble sort를 사용해 수행한다.

```
94 public void sortByDate() {
95     for (int i = 0; i < _numOrder-1; i++) {
96         String[] f1 = _orders[i].get();
97         for (int j = i+1; j < _numOrder; j++) {
98             String[] f2 = _orders[j].get();
99             if (f1[2].compareTo(f2[2]) > 0) { //
100                 Order tmp = _orders[i];
101                 _orders[i] = _orders[j];
102                 _orders[j] = tmp;
103             }
104         }
105     }
106 }
107
108 public void sortByCustomer() {
109     for (int i = 0; i < _numOrder-1; i++) {
110         String[] f1 = _orders[i].get();
111         for (int j = i+1; j < _numOrder; j++) {
112             String[] f2 = _orders[j].get();
113             if (f1[1].compareTo(f2[1]) > 0) { //
114                 Order tmp = _orders[i];
115                 _orders[i] = _orders[j];
116                 _orders[j] = tmp;
117             }
118         }
119     }
120 }
```

5) 그 외 일부 변수들의 명칭을 더 직관적인 이름으로 수정하였다.

2. OrderManagement의 수정

이 프로그램은 OrderListView와 ItemListView에서 setLayout을 통해 패널을 구성할 때 BorderLayout을 사용하였으며, 이 BorderLayout에서 주문을 하나하나 포함하고 있기 때문에 ArrayList를 별도로 사용할 필요가 없다. 설명하면, BorderLayout은 Container를 인자로 받으며 Container는 ArrayList로 구성되어 있고, add()를 통해 새 item을 추가할 수 있기 때문에 사실상 ArrayList를 사용한 것과 동일하다.

1) class 함수 내 변수 추가

Sort By Date/Customer와 Change/Done에 사용할 boolean 변수 _editMode를 정의한다. 이 변수는 ActionListener에서 false/true로 전환됨으로써 메인 화면 내 기능의 변환(Change/Done)을 하는 트리거가 되도록 한다. 또한 _editmode == true(Change 모드)일 경우 아래 MouseListener의 setEditMode()를 통해 다른 order를 선택하지 못하게 한다.

```
24 public class OrderManagement extends JFrame {
25     private JButton _add, _sort, _change, _save;
26     private OrderInfo _orderInput;
27     private OrderList _orderList;
28     private OrderListView _orderListView;
29     private boolean _sortByDate = false;
30     private boolean _editMode = false;
31     private static final String _defaultOrderPath = "파일 경로";
32 }
```

2) 마우스 동작 구동

별도의 class를 정의해 MouseListener의 인터페이스를 구현하여 마우스 이벤트를 처리할 수 있게 한다. class의 명칭은 SelectListener로 하며, 이를 OrdeerView와 ItemView에 연결하여 다중 item이 있는 order를 처리할 수 있게 하였다. 우선 현재 선택된 OrderView와 ItemView를 참조하는 변수와 Change의 모드를 체크하는 boolean 변수 _editMode를 정의한다.

기능 전체를 초기화하는 method reset()를 정의한다. 이 method에서는 참조 변수를 null로 설정하고 Change 변수의 값을 false로 하여 SelectListener를 초기화한다. 이를 통해 새로운 order가 추가되거나 sort로 순서가 변경될 때 호출되어 선택열(order)이 없는 상태로 한다.

마우스 클릭 동작을 제어하는 method setEditMode()를 정의한다. 이 method에서는 Change/Done 버튼이 눌릴 때 _editMode의 값을 인자로 호출하여 이 값이 true일 시 마우스 클릭을 처리하지 않도록 한다. 또한 OrderView와 ItemView의 setEditMode()를 통해 GUI 상의 수정 가능한 상태를 변경한다.

마우스 클릭 이벤트를 처리하는 method mouseClicked()는 _editmode가 true면 반응하지 않게 하여 change 상태에서 다른 order열의 선택을 방지한다. itemView를 클릭하면 해당 item을 _itemView의 parent에 해당하는 _orderView를 검색하고, orderView를 클릭하면 2개 이상 item이 있는 경우를 위해 가장 첫번째 _itemView를 선택하도록 한 후 orderView/itemView 각각의 setEditMode()를 통해 GUI의 상태를 변경한다. 또한 이전에 선택된 orderView/itemView는 각각 선택하지 않은 것으로 GUI 상태를 변경한다.

그 외 MouseListener의 다른 method들인 mousePressed/mouseReleased/mouseEntered/mouseExited는 구현하지 않고 명시만 한다.

```
315 private class SelectListener implements MouseListener {
316     public OrderView _orderView;
317     public ItemView _itemView;
318     public boolean _editMode = false;
319     public void reset() {
320         if (_itemView != null) _itemView.setSelected(false);
321         if (_orderView != null) _orderView.setSelected(false);
322         _itemView = null;
323         _orderView = null;
324     }
325     public void setEditMode(boolean mode) {
326         _editMode = mode;
327         if (_itemView != null) _itemView.setEditMode(mode);
328         if (_orderView != null) _orderView.setEditMode(mode);
329     }
330     public void mouseClicked(MouseEvent e) {
331         if (_editMode) return;
332         OrderView ov;
333         ItemView iv;
334         if (e.getSource() instanceof ItemView) {
335             iv = (ItemView)e.getSource();
336             ov = iv._itemListView._orderView;
337         } else {
338             ov = (OrderView)e.getSource();
339             iv = (ItemView)ov._itemListView.getComponent(0);
340         }
341         if (ov != _orderView) {
342             if (_orderView != null) _orderView.setSelected(false);
343             ov.setSelected(true);
344             _orderView = ov;
345         }
346         if (iv != _itemView) {
347             if (_itemView != null) _itemView.setSelected(false);
348             iv.setSelected(true);
349             _itemView = iv;
350         }
351     }
352     public void mousePressed(MouseEvent e) {}
353     public void mouseReleased(MouseEvent e) {}
354     public void mouseEntered(MouseEvent e) {}
355     public void mouseExited(MouseEvent e) {}
356 }
```

3) initWithFrame()에서 각 buttonPanel에 addActionListener를 통한 동작 부여

_add의 ActionListener에서는 OrderInfo의 setFrame()을 실행하도록 한다.

_sort의 ActionListener에서는 Sort 관련 boolean 변수 _sortByDate의 false/true 상태에 따라 각각 OrderList의 sortByDate()/sortByCustomer()를 실행시키고 변수 값을 true/false으로 바꾼다.

_change의 ActionListener에서는 위에서 정의한 Change 관련 boolean 변수 _editMode의 false/true

를 판별하여 그 값을 true/false로 바꾸고 아래 정의할 setEditMode()로 보낸다.

_save의 ActionListener에서는 OrderList의 saveOrders()를 실행하도록 하며, Exception의 여부에 따라 저장 완료 메시지를 표시하는 Dialog나 오류 메시지를 표시하는 Dialog를 띄우도록 한다.

```
66@    _add.addActionListener(new ActionListener() {
67@        @Override
68        public void actionPerformed(ActionEvent e) {
69            _orderInput.showFrame();
70        }
71    });
72@    _sort.addActionListener(new ActionListener() {
73@        @Override
74        public void actionPerformed(ActionEvent e) {
75            if (_sortByDate) {
76                _orderList.sortByCustomer();
77                _sortByDate = false;
78            } else {
79                _orderList.sortByDate();
80                _sortByDate = true;
81            }
82            _orderListView.updateOrders(_orderList);
83        }
84    });
85@    _change.addActionListener(new ActionListener() {
86@        @Override
87        public void actionPerformed(ActionEvent e) {
88            if (_editMode) {
89                _editMode = false;
90            } else {
91                _editMode = true;
92            }
93            _orderListView.setEditMode(_editMode);
94        }
95    });
96@    _save.addActionListener(new ActionListener() {
97@        @Override
98        public void actionPerformed(ActionEvent e) {
99            try {
100                _orderList.saveOrders(_defaultOrderPath);
101                JOptionPane.showMessageDialog(null, "File Saved");
102            } catch (Exception e1) {
103                JOptionPane.showMessageDialog(null, "Error Occured");
104            }
105        }
106    });
```

4) addOrder()의 수정

OrderList의 addOrder()에서 정보를 받고 OrderListView의 updateOrders()를 통해 리스트를 업데이트 하는 기존의 동작을 유지하되, try-catch를 통해 에러가 발생했을 시 콘솔에 이를 출력하도록 한다.

```
112@    public void addOrder(String order) {
113        try {
114            _orderList.addOrder(order);
115            _orderListView.updateOrders(_orderList);
116        } catch (Exception e) {
117            System.out.println("ERROR: incorrect order");
118        }
119    }
```

5) class OrderListView의 수정

먼저 SelectListener를 참조하는 변수를 정의하며, updateOrders()가 수행될 시 마우스 선택을 초기화 한다. 또한 SelectListener의 setEditMode를 실행하는 method를 추가로 정의함으로써 Change에서 이를 실행할 수 있도록 한다.

```
161@    public void updateOrders(OrderList orderList) {
162        JPanel data = new JPanel();
163        data.setLayout(new BoxLayout(data, BoxLayout.Y_AXIS));
164        for (int i = 0; i < orderList.numOrders(); i++)
165            data.add(new OrderView(this, orderList.getOrder(i)));
166        remove(_layout.getLayoutComponent(BorderLayout.CENTER));
167        add(data, BorderLayout.CENTER);
168
169        _selector.reset();
170        revalidate();
171    }
```

6) class OrderView의 수정

우선 Order, OrderListView, ItemListView, JTextField를 참조하는 변수를 정의하며 Order, OrderListView는 생성자를 통해 전달된다.

Order 변수는 현재 목록에 표시할 주문 정보를 저장하고 있는 content이며, OrderListView와 ItemListView 변수는 각각 OrderView의 parent와 child에 해당하며 주문/상품 목록과의 상호작용을 위해 사용된다. JTextField 변수는 주문의 주소를 편집하고 표시하는데 사용된다.

또한 선택되었을 시 해당 주문 단락의 배경색이자 선택 여부를 변경하는 method인 setSelected()와, 각 주문이 편집 가능한 상태인지 확인하고 변경이 완료된 상태면 JTextField _textField에 작성된 String을 가져와 Order의 주소를 변경하는 method setEditMode()를 정의한다.

```
181
182 private class OrderView extends JPanel {
183     public Order _order;
184     public OrderListView _orderListView;
185     public ItemListView _itemListView;
186     private JTextField _textField;
187
188     public OrderView(OrderListView parent, Order order) {
189         _order = order;
190         _orderListView = parent;
191         addMouseListener(_orderListView._selector);
192
193         double[] colRatio = {0.12, 0.12, 0.12, 0.12, 0.4, 0.12};
194         setLayout(new RatioGridLayout(1, 6, colRatio));
195
196         String[] fields = order.get();
197         add(new JLabel(" " + fields[0], JLabel.LEFT));
198         add(new JLabel(fields[1]));
199         add(new JLabel(fields[2]));
200         add(new JLabel(fields[3] + " ", JLabel.RIGHT));
201
202         _itemListView = new ItemListView(this, order);
203         add(_itemListView);
204
205         _textField = new JTextField(fields[4]);
206         _textField.setBorder(new LineBorder(Color.black, 1));
207         _textField.setEditable(false);
208         add(_textField);
209     }
210
211     public void setEditMode(boolean mode) {
212         _textField.setEditable(mode);
213         if (!mode)
214             _order.setAddress(_textField.getText());
215     }
216 }
```

7) class ItemListView의 수정

타 class와의 용어 통일을 위해 명칭을 ProductListView에서 ItemListView로 변경하였다. 그리고 OrderView를 참조하는 변수를 정의하며, 이와 Order 객체를 통해 새로 구성요소를 배치하고 주문에 속한 각 상품의 정보를 가져와 패널에 추가한다.

```
222 private class ItemListView extends JPanel {
223     public OrderView _orderView;
224
225     public ItemListView(OrderView parent, Order order) {
226         _orderView = parent;
227         setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
228         for (int i = 0; i < order.getItemCount(); i++) {
229             String[] fields = order.getItem(i);
230             if (fields == null) break;
231             add(new ItemView(this, fields, i));
232         }
233     }
234 }
```

8) class ItemView의 수정

타 class와의 용어 통일을 위해 명칭을 ProductView에서 ItemView로 변경하였다. 그리고 index값, ItemListView, JComboBox를 참조하는 변수를 정의하며, 이는 ItemView의 생성자를 통해 외부에서 값을 받아 정보와 ComboBox를 패널에 추가한다.

또한 OrderView에서 추가한 것과 동일하게, 선택되었을 시 해당 주문 단락의 색을 변경하는 method인 setSelected()와, 각 주문이 편집 가능한 상태인지 확인하고 가능하지 않다면 Order의 setItemStatus()를 불러와 현재 String에 작성되어 있는 주소로 전달하는 method setEditMode()를 정의한다.

```
232 private class ItemView extends JPanel {
233     public int _index;
234     public ItemListView _itemListView;
235     public JComboBox _comboBox;
236
237     public ItemView(ItemListView parent, String[] fields, int idx) {
238         _index = idx;
239         _itemListView = parent;
240         addMouseListener(_itemListView._orderView._orderListView._selector);
241         setLayout(new GridLayout(1, 4));
242         setBorder(new LineBorder(Color.black, 1));
243
244         Color c = new Color(0,0,0,0);
245         add(new JLabel(fields[0], c));
246         add(new JLabel(fields[1], c));
247         add(new JLabel(fields[2], c));
248         String[] listStatus = {"CANCELED", "PREPARING", "SHIPPED", "DELIVERED", "RETURNED"};
249         _comboBox = new JComboBox(listStatus);
250         _comboBox.setSelectedItem(fields[3]);
251         _comboBox.setEnabled(false);
252         add(_comboBox);
253     }
254
255     public void setEditMode(boolean mode) {
256         _comboBox.setEnabled(mode);
257         if (!mode) {
258             Order order = _itemListView._orderView._order;
259             order.setItemStatus(_index, (String)_comboBox.getSelectedItem());
260         }
261     }
262 }
263
```

3. OrderInfo의 수정

1) OrderInfo()의 수정

INPUT 버튼이 동작하도록 ActionListener를 사용해 동작을 추가한다. actionPerformed 내부에서는 창을 종료하고 OrderManagement의 addOrder()를 통해 작성한 새 주문을 추가하도록 한다.

그 외 패널을 명시하는 p를 inputPanel로 교체하는 등 일부 변수들의 명칭을 더 직관적인 이름으로 수정하였다.

```
--
45     _input = new JButton("INPUT");
46     _input.addActionListener(new ActionListener() {
47         @Override
48         public void actionPerformed(ActionEvent e) {
49             setVisible(false);
50             orderManager.addOrder(get());
51         }
52     });
53     add(_input);
```

2) TitleTextField()의 수정

파일의 텍스트를 반환하는 method getText()를 정의한다. 이는 OrderManagement/OrderView의 setEditMode()에서 입력된 주소를 전달하기 위함이다.

4. 디버깅

- 1) 특정 주문을 선택하지 않고 Change 버튼을 누르면 항목을 선택할 수 없게 되며, 이는 Change 버튼을 다시 누른 뒤에 정상적으로 작동하는 현상을 확인하였다.

이를 해결하기 위해서는 주문이 선택되었는지 true/false로 확인해주는 boolean 요소가 필요하며, 선택이 true인 경우에만 Change 버튼을 실행하는 동작을 수행해야 한다.

우선 SelectListener 내부에 OrderView가 선택되었을 시 true를 반환하고, 그렇지 않으면 false를 반환하는 method isSelected()를 정의한다.

```
337 public boolean isSelected() {  
338     return (_orderView != null) ? true : false;  
339 }
```

그 다음 특정 주문을 표시하는 가장 상위 패널인 OrderListView 내부에 선택되었는지 여부를 확인하여 SelectListener의 isSelected 값을 반환하는 method isSelected()를 정의한다.

```
177 public boolean isSelected() {  
178     return _selector.isSelected();  
179 }
```

마지막으로 initFrame의 Change 버튼을 담당하는 ActionListener에서 OrderListView의 isSelected()가 true를 반환해야만 _editMode가 true로 변하도록 한다.

```
85 _change.addActionListener(new ActionListener() {  
86     @Override  
87     public void actionPerformed(ActionEvent e) {  
88         if (_editMode) {  
89             _editMode = false;  
90         } else if (_orderListView.isSelected()) {  
91             _editMode = true;  
92         }  
93         _orderListView.setEditMode(_editMode);  
94     }  
95 });
```

이를 통해 특정 주문을 선택하지 않고 Change 버튼을 눌러도 항목을 선택할 수 있게 되었다.

- 2) 프로그램을 실행시켰을 때 Ship Address를 제외한 다른 주문 패널들의 색이 조건에서 제시된 예제와 색이 다른 것을 확인하였다.

Order ID	Buyer	Date	(Total) Amount	Item(s)				Ship
				ID	Name	Price	Status	
20230604162408	CAU	2023-06-04_16:24	100	C1111	CAU	100	PREPARING	CAU
202302101310001	Kim	2023-02-10_13:10	30000	2005	T-Shirt	20000	DELIVERED	hardware
				1001	Apple	10000	DELIVERED	
202301101730010	Lee	2023-01-10_17:30	5000	3001	Pencil	5000	DELIVERED	Seoul
202301030910010	Park	2023-01-03_09:10	10000	1001	Apple	10000	SHIPPED	CAU

이는 주문 패널에서 `setEnabled`를 사용하였기 때문에 패널의 기본 색으로 고정된 것이며, 이를 해결하기 위해서는 이 패널들에 강제로 색을 입혀야 한다.

우선 `OrderView`에서 패널의 색을 `Color`을 사용해 예제와 같은 (238, 238, 238)로 고정하며, method `setSelected()`를 정의해 선택되면 하얀 색으로 바뀌게 한다.

```

195         setBackground(new Color(238, 238, 238));
196
197         String[] fields = order.get();
198         add(new JLabel(" " + fields[0], JLabel.LEFT));
199         add(new JLabel(fields[1]));
200         add(new JLabel(fields[2]));
201         add(new JLabel(fields[3] + " ", JLabel.RIGHT));
202
203         _itemListView = new ItemListView(this, order);
204         add(_itemListView);
205
206         _textField = new JTextField(fields[4]);
207         _textField.setBorder(new LineBorder(Color.black, 1));
208         _textField.setEditable(false);
209         add(_textField);
210     }
211
212     public void setSelected(boolean select) {
213         if (select)
214             setBackground(Color.white);
215         else
216             setBackground(new Color(238, 238, 238));
217     }

```

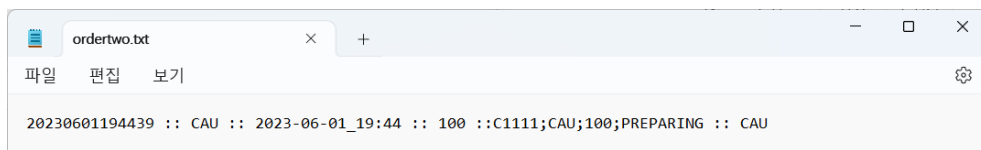
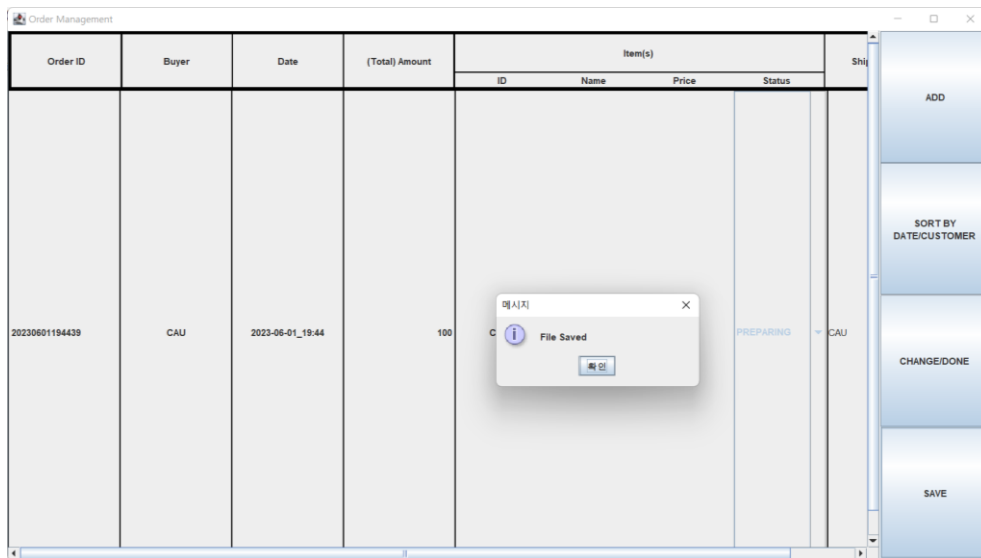
`ItemView`에서도 똑같이 색을 고정하고 선택되었을 때만 바뀌게 한다.

```

251         setBackground(new Color(238, 238, 238));
252
253         Color c = new Color(0,0,0,0);
254         add(new JLabel(fields[0], c));
255         add(new JLabel(fields[1], c));
256         add(new JLabel(fields[2], c));
257         String[] listStatus = {"CANCELED", "PREPARING", "SHIPPED", "DELIVERED", "RETURNED"};
258         _comboBox = new JComboBox(listStatus);
259         _comboBox.setSelectedItem(fields[3]);
260         _comboBox.setEnabled(false);
261         add(_comboBox);
262     }
263
264     public void setSelected(boolean select) {
265         if (select)
266             setBackground(Color.white);
267         else
268             setBackground(new Color(238, 238, 238));
269     }

```

이를 통해 패널의 색을 통일시킬 수 있었다.



이 구동을 통해 새로운 파일 ordertwo.txt가 생성되었음을 확인할 수 있었다.

2. ADD 기능: 모든 내용이 공백인 주문을 추가

Input Order Information

Byuer Name	Item ID	Item Name	Item Price
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Shipping Address	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

INPUT

Order ID	Buyer	Date	(Total) Amount	Item(s)				Shi
				ID	Name	Price	Status	
202301030910010	Park	2023-01-03_09:10	10000	1001	Apple	10000	SHIPPED	CAU
202301101730010	Lee	2023-01-10_17:30	5000	3001	Pencil	5000	DELIVERED	Seoul
202302101310001	Kim	2023-02-10_13:10	30000	2005	T-Shirt	20000	CANCELED	Dept. Soft
				1001	Apple	10000	DELIVERED	

이 구동을 통해 내용이 존재하지 않는 주문은 의도한 대로 추가되지 않음을 확인할 수 있었다.

3. ADD 기능: 가격이 정수가 아닌 주문을 추가(CAU-CAU-CAU-CAU-CAU)

Input Order Information

Byuer Name	Item ID	Item Name	Item Price
CAU	<input type="text"/>	<input type="text"/>	<input type="text"/>
Shipping Address	CAU	CAU	CAU
CAU	<input type="text"/>	<input type="text"/>	<input type="text"/>

INPUT

Order ID	Buyer	Date	(Total) Amount	Item(s)				Ship
				ID	Name	Price	Status	
202301030910010	Park	2023-01-03_09:10	10000	1001	Apple	10000	SHIPPED	CAU
202301101730010	Lee	2023-01-10_17:30	5000	3001	Pencil	5000	DELIVERED	Seoul
202302101310001	Kim	2023-02-10_13:10	30000	2005	T-Shirt	20000	CANCELED	Dept. Soft
				1001	Apple	10000	DELIVERED	

이 구동을 통해 내용이 잘못된 주문은 추가되지 않음을 확인할 수 있었다.

2) 최종 실행 결과

1. 실행 화면(기본)

Order ID	Buyer	Date	(Total) Amount	Item(s)				Ship Address
				ID	Name	Price	Status	
202302101310001	Kim	2023-02-10_13:10	30000	2005	T-Shirt	20000	CANCELED	Dept. Software
				1001	Apple	10000	DELIVERED	
202301101730010	Lee	2023-01-10_17:30	5000	3001	Pencil	5000	DELIVERED	Seoul
202301030910010	Park	2023-01-03_09:10	10000	1001	Apple	10000	SHIPPED	CAU

```

order.txt
파일 편집 보기

202301030910010 :: Park :: 2023-01-03_09:10 :: 10000 ::1001;Apple;10000;SHIPPED :: CAU
202301101730010 :: Lee :: 2023-01-10_17:30 :: 5000 ::3001;Pencil;5000;DELIVERED :: Seoul
202302101310001 :: Kim :: 2023-02-10_13:10 :: 30000 ::2005;T-Shirt;20000;CANCELED :
1001;Apple;10000;DELIVERED :: Dept.Software
  
```

2. ADD 기능(새 정보 CAU-CAU-C1111-CAU-100을 입력)

Byuer Name	Item ID	Item Name	Item Price	INPUT
CAU				
Shipping Address	C1111	CAU	100	
CAU				

Order ID	Buyer	Date	(Total) Amount	Item(s)				Ship Address
				ID	Name	Price	Status	
202302101310001	Kim	2023-02-10_13:10	30000	2005	T-Shirt	20000	CANCELED	Dept. Software
				1001	Apple	10000	DELIVERED	
202301101730010	Lee	2023-01-10_17:30	5000	3001	Pencil	5000	DELIVERED	Seoul
202301030910010	Park	2023-01-03_09:10	10000	1001	Apple	10000	SHIPPED	CAU
20230531211703	CAU	2023-05-31_21:17	100	C1111	CAU	100	PREPARING	CAU

3. SORT 기능

초기 상태(Kim->Lee->Park->CAU)

Order ID	Buyer	Date	(Total) Amount	Item(s)				Ship Address
				ID	Name	Price	Status	
202302101310001	Kim	2023-02-10_13:10	30000	2005	T-Shirt	20000	CANCELED	Dept. Software
				1001	Apple	10000	DELIVERED	
202301101730010	Lee	2023-01-10_17:30	5000	3001	Pencil	5000	DELIVERED	Seoul
202301030910010	Park	2023-01-03_09:10	10000	1001	Apple	10000	SHIPPED	CAU
20230531211703	CAU	2023-05-31_21:17	100	C1111	CAU	100	PREPARING	CAU

SORT 버튼을 한 번 눌러 DATE 순서로 정렬된 상태(Park->Lee->Kim->CAU)

Order ID	Buyer	Date	(Total) Amount	Item(s)				Ship Address
				ID	Name	Price	Status	
202301030910010	Park	2023-01-03_09:10	10000	1001	Apple	10000	SHIPPED	CAU
202301101730010	Lee	2023-01-10_17:30	5000	3001	Pencil	5000	DELIVERED	Seoul
202302101310001	Kim	2023-02-10_13:10	30000	2005	T-Shirt	20000	DELIVERED	Hardware
				1001	Apple	10000	DELIVERED	
20230604162408	CAU	2023-06-04_16:24	100	C1111	CAU	100	PREPARING	CAU

SORT 버튼을 두 번 눌러 CUSTOMER 순서로 정렬된 상태(CAU->Kim->Lee->Park)

Order ID	Buyer	Date	(Total) Amount	Item(s)				Ship Address
				ID	Name	Price	Status	
20230531211703	CAU	2023-05-31_21:17	100	C1111	CAU	100	PREPARING	CAU
202302101310001	Kim	2023-02-10_13:10	30000	2005	T-Shirt	20000	CANCELED	Dept. Software
				1001	Apple	10000	DELIVERED	
202301101730010	Lee	2023-01-10_17:30	5000	3001	Pencil	5000	DELIVERED	Seoul
202301030910010	Park	2023-01-03_09:10	10000	1001	Apple	10000	SHIPPED	CAU

4. CHANGE 기능(Kim의 첫번째 상품 정보를 CANCELED/Dept. Software->DELIVERED/Hardware Dept로 변경)

CHANGE 버튼은 주문을 선택하지 않으면 작동하지 않으며, 주문을 선택하고 눌러야만 작동한다.

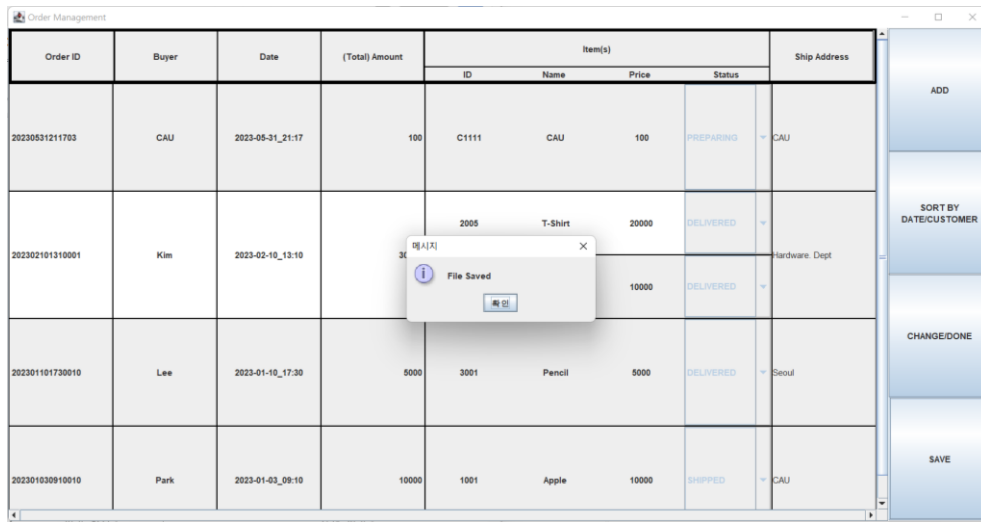
아래 화면은 주문 하나를 선택하고 CHANGE 버튼을 누른 상태로, 이 상태에서는 해당 주문 패널의 색이 하얀색으로 바뀌며 Status와 Ship Address를 변경할 수 있다.

Order ID	Buyer	Date	(Total) Amount	Item(s)				Ship Address
				ID	Name	Price	Status	
20230531211703	CAU	2023-05-31_21:17	100	C1111	CAU	100	PREPARING	CAU
202302101310001	Kim	2023-02-10_13:10	30000	2005	T-Shirt	20000	CANCELED	Dept. Software
				1001	Apple	10000	DELIVERED	
202301101730010	Lee	2023-01-10_17:30	5000	3001	Pencil	5000	DELIVERED	Seoul
202301030910010	Park	2023-01-03_09:10	10000	1001	Apple	10000	SHIPPED	CAU

위의 화면에서 정보를 수정하고 CHANGE 버튼을 다시 누르면 아래 화면과 같이 정보는 수정된 상태로 저장된다.

Order ID	Buyer	Date	(Total) Amount	Item(s)				Ship Address
				ID	Name	Price	Status	
20230531211703	CAU	2023-05-31_21:17	100	C1111	CAU	100	PREPARING	CAU
202302101310001	Kim	2023-02-10_13:10	30000	2005	T-Shirt	20000	DELIVERED	Hardware Dept
				1001	Apple	10000	DELIVERED	
202301101730010	Lee	2023-01-10_17:30	5000	3001	Pencil	5000	DELIVERED	Seoul
202301030910010	Park	2023-01-03_09:10	10000	1001	Apple	10000	SHIPPED	CAU

5. SAVE 기능(지금까지 수행한 모든 변경 사항을 저장)



The screenshot shows the 'Order Management' application window. It contains a table with columns: Order ID, Buyer, Date, (Total) Amount, Item(s) (subdivided into ID, Name, Price, Status), and Ship Address. The table lists four orders. A 'File Saved' dialog box is overlaid on the table, indicating that the data has been saved successfully. The dialog box has a blue '확인' (OK) button.

Order ID	Buyer	Date	(Total) Amount	Item(s)				Ship Address
				ID	Name	Price	Status	
20230531211703	CAU	2023-05-31_21:17	100	C1111	CAU	100	PREPARING	CAU
202302101310001	Kim	2023-02-10_13:10	30000	2005	T-Shirt	20000	DELIVERED	Hardware. Dept
202301101730010	Lee	2023-01-10_17:30	5000	3001	Pencil	5000	DELIVERED	Seoul
202301030910010	Park	2023-01-03_09:10	10000	1001	Apple	10000	SHIPPED	CAU

order.txt

```
20230604162408 :: CAU :: 2023-06-04_16:24 :: 100 :: C1111;CAU;100;PREPARING :: CAU
202302101310001 :: Kim :: 2023-02-10_13:10 :: 30000 :: 2005;T-Shirt;20000;DELIVERED :
1001;Apple;10000;DELIVERED :: Hardware. Dept
202301101730010 :: Lee :: 2023-01-10_17:30 :: 5000 :: 3001;Pencil;5000;DELIVERED :: Seoul
202301030910010 :: Park :: 2023-01-03_09:10 :: 10000 :: 1001;Apple;10000;SHIPPED :: CAU
```

6. 평가표

평가 항목	자체 평가
<p>완성도 (동작 여부)</p> <ul style="list-style-type: none"> - "초기"동작(주문 파일 읽기) - "Add" 동작 (주문 리스트 갱신) - "Sort" 동작 - "Change" 동작 (선택 조건에 따른 동작 확인) - "Save" 동작 (Dialogue 방식) - 기타 비정상 동작 실험 (즉, 구현 한계 점검) 	<p>이 프로그램은 외부의 주문 파일을 읽어 들이는 초기 동작이 수행되었다.</p> <p>추가 주문을 하나씩 받아들여 주문 리스트를 갱신하는 Add 동작이 수행되었다.</p> <p>첫 버튼 클릭에서 날짜 순서대로 정렬되고, 두번째 버튼 클릭에서 주문자 이름 순서대로 정렬을 하는 Sort 동작이 수행되었다.</p> <p>임의의 주문 패널 하나를 클릭한 다음 첫 버튼 클릭에서 정보 수정이 가능해지고, 두번째 버튼 클릭에서 정보 수정을 완료하는 Change/Done 동작이 수행되었다.</p> <p>수정된 파일을 저장하고 Dialogue를 출력하여 파일이 저장되었다는 것을 확인해주는 Save 동작이 수행되었다.</p> <p>존재하지 않는 파일의 경로로 실행할 시 주문이 없는 상태로 패널이 출력되며, 이때 주문을 추가하고 저장하면 새로운 파일이 생성되는 것을 확인하였다.</p> <p>Add를 수행할 때 아무것도 입력하지 않거나 잘못된 값을 입력하여 추가할 시 Order의 set()에 추가한 조건에 따라 빈 정보가 목록에 더해지지 않는 것을 확인하였다.</p>
<p>설계 노트</p> <ul style="list-style-type: none"> - 주요 결정사항 및 근거 - 한계/문제점 	<p>이 프로그램의 구조는 GUI를 통해 작업을 수행하며, GUI의 구조는 하나의 큰 패널을 여러 개의 작은 패널로 분해하여 각각을 하위 class로 작성하였다. 그리고 하위 class에서 외부 class의 동작을 넘겨받고 이를</p>

<ul style="list-style-type: none"> - 해결 방안 - 필수내용: * 프로그램 구성(Class 구조) * member visibility 	<p>상위 class에서 다시 받아 재구성하도록 하였다.</p> <p>라이브러리에 존재하는 기존 프레임 관련 class들은 패널의 크기를 자동으로 조정해버린다. 그러므로 패널의 크기 및 간격을 임의로 설정하기 위해 외부에서 RatioGridLayout을 참고하여 사용하였다.</p> <p>위에서도 설명했듯이 ArrayList는 명시적으로 사용하지 않았으며, 이는 패널을 구성할 때 사용한 BoxLayout가 이를 모두 포함하고 있기 때문이다.</p> <p>주문 패널에 setEnabled를 사용하였기 때문에 패널의 기본 색이 변하였다. 그러므로 색을 일치시키기 위해서 패널의 색을 특정값으로 고정시키고, 선택시에만 변하게 하였다.</p> <p>프로그램 class의 구성요소 중 다른 class에서 사용되어야 할 일부 변수나 생성자, method를 제외한 나머지를 private으로 정의하였다.</p> <p>프로그램을 통해서는 조건 사항에 같이 제시되었던 복수 상품의 정보 입력이 불가능하며, 텍스트를 직접 수정해야만 입력이 가능하다.</p> <p>특정 항목을 선택하지 않고 Change 버튼을 누르면 항목을 선택할 수 없게 되며, 이는 Change 버튼을 다시 누른 뒤에 정상적으로 작동된다.</p>
<p>리포트</p> <ul style="list-style-type: none"> - 평가자 시각으로 리포트 검토 - 위의 평가 요소들이 명확하게 기술되었는가? 	<p>이 프로그램은 정상적으로 작동하는 동시에 제시된 요구사항을 만족하는 동시에 대부분 수업시간에서 설명된 class로 작성되었다.</p>
<p>총평</p>	<p>종합적으로 이 프로그램의 알고리즘은 다른 프로그램에 적용이 힘들고 오직 이 패널을 구현하기 위한 경직된 구조로 이루어져 있다.</p>

7. 소스 코드

<pre> Order.java package default_package; import java.io.PrintStream; import java.text.ParseException; import java.text.SimpleDateFormat; import java.time.LocalDateTime; import java.time.format.DateTimeFormatter; import java.util.Date; public class Order { private final int MAX_ITEM = 100; private int _numItem = 0; private String _id; private String _name; private String _time; private int _price; private Item[] _items = new Item[MAX_ITEM]; private String _location; private class TimeException extends Exception {} </pre>
--


```

private class ItemException extends Exception {}

Order(String[] fields) throws Exception {
    set(fields);
}

Order(String str) throws Exception {
    String[] tok = str.split("::");
    for (int i = 0; i < tok.length; i++) tok[i] = tok[i].trim();
    try {
        set(tok);
    } catch(ItemException e) {
        System.out.println("        -- No ordered items -- invalid
Order info line: " + str);
        throw new Exception();
    } catch(TimeException e) {
        System.out.println("        -- No time order is issued --
invalid Order info line: " + str);
        throw new Exception();
    }
}

public void print() {
    print(System.out);
}

public void print(PrintStream ps) {
    ps.printf("%s :: %s :: %s :: %d ::", _id, _name, _time, _price);
    for (int i = 0; i < _numItem; i++) {
        _items[i].print(ps);
        if (i < _numItem-1) ps.printf(" : ");
    }
    ps.printf(" :: %s\n", _location);
}

public String[] get() {
    String[] fields = new String[5];
    fields[0] = _id;
    fields[1] = _name;
    fields[2] = _time;
    fields[3] = String.valueOf(_price);
    fields[4] = _location;
    return fields;
}

public String[] getItem(int i) {
    if (i < _numItem) {
        String[] fields = new String[4];
        fields[0] = _items[i]._id;
        fields[1] = _items[i]._name;
        fields[2] = String.valueOf(_items[i]._price);
        fields[3] = _items[i]._status;
        return fields;
    }
    return null;
}

public void setAddress(String addr) {
    _location = addr;
}

public void setItemStatus(int i, String status) {
    if (i < _numItem) _items[i]._status = status;
}

```

```

private void set(String[] fields) throws Exception {
    String[] tok = fields[4].split(":");
    _numItem = 0;
    for (int i = 0; i < tok.length; i++) {
        _items[i] = new Item(tok[i]);
        _numItem++;
    }

    LocalDateTime now = LocalDateTime.now();
    if (fields[0] == "" || fields[0].length() == 0) {
        _id = now.format(DateTimeFormatter.ofPattern("yyyyMMddHHmmss"));
    } else {
        _id = fields[0];
    }

    _name = fields[1];

    if (fields[2] == "" || fields[2].length() == 0) {
        _time = now.format(DateTimeFormatter.ofPattern("yyyy-MM-
dd_HH:mm"));
    } else {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd_HH:mm");
        sdf.setLenient(false);
        try {
            Date date = sdf.parse(fields[2]);
            _time = fields[2];
        } catch (ParseException e) {
            System.out.println("Date Format Error -- " +
fields[2]);
            throw new TimeException();
        }
    }

    int price_sum = 0;
    for (int i = 0; i < _numItem; i++) {
        price_sum += _items[i]._price;
    }
    if (fields[3] == "" || fields[3].length() == 0) {
        _price = price_sum;
    } else {
        _price = Integer.parseInt(fields[3]);
        if (_price != price_sum) {
            System.out.println("ERROR: price mismatched");
        }
    }

    _location = fields[5];
}

private class Item {
    public String _id;
    public String _name;
    public int _price;
    public String _status;

    Item(String str) throws Exception {
        String[] tok = str.split(";");
        if (tok.length != 4) {
            System.out.println("wrong ordered-item description format
-- " + str);
            throw new ItemException();
        }

        for (int i = 0; i < tok.length; i++) tok[i] = tok[i].trim();
        _id = tok[0];
    }
}

```

```

        _name = tok[1];
        _price = Integer.parseInt(tok[2]);
        _status = tok[3];
    }

    public void print(PrintStream ps) {
        ps.printf("%s;%s;%d;%s", _id, _name, _price, _status);
    }
}

```

OrderList.java

```

package default_package;

import java.io.File;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.io.PrintStream;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Scanner;

public class OrderList {
    private final int MAX_NUM = 100;
    private int _numOrder = 0;
    private Order[] _orders = new Order[MAX_NUM];

    OrderList() {}

    OrderList(String orderFileName) {
        loadOrders(orderFileName);
    }

    public void print() {
        for (int i = 0; i < _numOrder; i++) {
            _orders[i].print(System.out);
        }
    }

    public void print(PrintStream ps) {
        for (int i = 0; i < _numOrder; i++) {
            _orders[i].print(ps);
        }
    }

    public int numOrders() {
        return _numOrder;
    }

    public Order getOrder(int i) {
        return _orders[i];
    }

    public void addOrder(String order) {
        try {
            _orders[_numOrder] = new Order(order);
            _numOrder++;
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}

```

```

public void loadOrders(String orderFileName) {
    File file = new File(orderFileName);
    Scanner input;
    try {
        input = new Scanner(file);
    } catch (Exception e) {
        System.out.println("Unknwon OrderList data File");
        return;
    }
    while (input.hasNext()) {
        String line = input.nextLine();
        line = line.trim();
        if (line.length() > 0 &&
            !(line.charAt(0) == '/' && line.charAt(1) == '/')) {
            try {
                _orders[_numOrder] = new Order(line);
                _numOrder++;
            } catch (Exception e) {
                _numOrder = 0;
                return;
            }
        }
    }
}

public void saveOrders(String orderFileName){
    try {
        File file = new File(orderFileName);
        if (!file.exists()) {
            file.createNewFile();
        }
        print(new PrintStream(file));
    } catch (Exception e) {
        System.out.println(e.getMessage());
        throw new Exception();
    }
}

public void sortByDate() {
    for (int i = 0; i < _numOrder-1; i++) {
        String[] f1 = _orders[i].get();
        for (int j = i+1; j < _numOrder; j++) {
            String[] f2 = _orders[j].get();
            if (f1[2].compareTo(f2[2]) > 0) { // compare date fields
                Order tmp = _orders[i];
                _orders[i] = _orders[j];
                _orders[j] = tmp;
            }
        }
    }
}

public void sortByCustomer() {
    for (int i = 0; i < _numOrder-1; i++) {
        String[] f1 = _orders[i].get();
        for (int j = i+1; j < _numOrder; j++) {
            String[] f2 = _orders[j].get();
            if (f1[1].compareTo(f2[1]) > 0) { // compare name fields
                Order tmp = _orders[i];
                _orders[i] = _orders[j];
                _orders[j] = tmp;
            }
        }
    }
}

```

```
}  
}
```

RatioGridLayout.java

```
package default_package;  
  
import java.awt.*;  
import javax.swing.*;  
  
// https://www.phind.com/search?cache=f2909367-8739-4879-a500-1924a16b63b8  
public class RatioGridLayout extends GridLayout {  
    private final double[] _colRatio;  
  
    RatioGridLayout(int rows, int cols, double[] colRatio) {  
        super(rows, cols);  
        _colRatio = colRatio;  
    }  
  
    @Override  
    public void layoutContainer(Container parent) {  
        int components = parent.getComponentCount();  
        if (components == 0) return;  
  
        Insets insets = parent.getInsets();  
        int totalWidth = parent.getWidth() - insets.left - insets.right;  
        int totalHeight = parent.getHeight() - insets.top - insets.bottom;  
        int cols = getColumns();  
        int rows = components / cols;  
        if (components % cols > 0) rows++;  
  
        int[] colWidths = new int[cols];  
        for (int i = 0; i < cols; i++)  
            colWidths[i] = (int) (totalWidth * _colRatio[i]);  
  
        int x = insets.left;  
        int y = insets.top;  
        for (int r = 0; r < rows; r++) {  
            for (int c = 0; c < cols; c++) {  
                int i = r * cols + c;  
                if (i < components) {  
                    Component component = parent.getComponent(i);  
                    component.setBounds(x, y, colWidths[c], totalHeight / rows);  
                    x += colWidths[c];  
                }  
            }  
            y += totalHeight / rows;  
            x = insets.left;  
        }  
    }  
}
```

OrderManagement.java

```
package default_package;  
  
import java.awt.BorderLayout;  
import java.awt.Color;  
import java.awt.Dimension;  
import java.awt.GridLayout;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.MouseEvent;  
import java.awt.event.MouseListener;  
import java.util.ArrayList;  
  
import javax.swing.BoxLayout;
```

```

import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextField;
import javax.swing.border.LineBorder;

public class OrderManagement extends JFrame {
    private JButton _add, _sort, _change, _save;
    private OrderInfo _orderInput;
    private OrderList _orderList;
    private OrderListView _orderListView;
    private boolean _sortByDate = false;
    private boolean _editMode = false;
    private static final String _defaultOrderPath = "파일 경로";

    public OrderManagement() {
        _orderList = new OrderList();
        initFrame();
    }

    public OrderManagement(String orderFileName) {
        _orderList = new OrderList(orderFileName);
        initFrame();
    }

    private void initFrame() {
        setTitle("Order Management");
        setSize(1280, 720);
        setLayout(new BorderLayout());

        _orderInput = new OrderInfo(this);
        _orderListView = new OrderListView(_orderList);
        JScrollPane scroll = new JScrollPane(_orderListView);

        scroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        add(scroll, BorderLayout.CENTER);

        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new GridLayout(4, 1));
        _add = new JButton("ADD");
        _sort = new JButton("<html><center>SORT  
BY<br>DATE/CUSTOMER</center></html>");
        _change = new JButton("CHANGE/DONE");
        _save = new JButton("SAVE");
        buttonPanel.add(_add);
        buttonPanel.add(_sort);
        buttonPanel.add(_change);
        buttonPanel.add(_save);
        add(buttonPanel, BorderLayout.EAST);

        _add.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                _orderInput.showFrame();
            }
        });
        _sort.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (_sortByDate) {
                    _orderList.sortByCustomer();
                }
            }
        });
    }
}

```

```

        _sortByDate = false;
    } else {
        _orderList.sortByDate();
        _sortByDate = true;
    }
    _orderListView.updateOrders(_orderList);
}
});
_change.addActionListener(new ActionListener( ) {
@Override
public void actionPerformed(ActionEvent e) {
    if (_editMode) {
        _editMode = false;
    } else if (_orderListView.isSelected()) {
        _editMode = true;
    }
    _orderListView.setEditMode(_editMode);
}
});
_save.addActionListener(new ActionListener( ) {
@Override
public void actionPerformed(ActionEvent e) {
    try {
        _orderList.saveOrders(_defaultOrderPath);
        JOptionPane.showMessageDialog(null, "File
Saved");
    } catch (Exception e1) {
        JOptionPane.showMessageDialog(null, "Error
Occured");
    }
}
});

setVisible(true);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

public void addOrder(String order) {
    try {
        _orderList.addOrder(order);
        _orderListView.updateOrders(_orderList);
    } catch (Exception e) {
        System.out.println("ERROR: incorrect order");
    }
}

private class OrderListView extends JPanel {
    private SelectListener _selector = new SelectListener();
    private BorderLayout _layout = new BorderLayout();

    public OrderListView(OrderList orderList) {
        setLayout(_layout);
        setPreferredSize(new Dimension(1200, 700));

        //title
        JPanel titleItemSub = new JPanel();
        titleItemSub.setLayout(new GridLayout(1, 4));
        Color c = new Color(0,0,0,0);
        titleItemSub.add(new TextLabel("ID", c));
        titleItemSub.add(new TextLabel("Name", c));
        titleItemSub.add(new TextLabel("Price", c));
        titleItemSub.add(new TextLabel("Status", c));
        titleItemSub.setBorder(new LineBorder(Color.black, 1));

        JPanel titleItem = new JPanel();

```

```

        titleItem.setLayout(new BorderLayout());
        titleItem.add(new
TextLabel("<html><br>Item(s)<br><br></html>"), BorderLayout.CENTER);
        titleItem.add(titleItemSub, BorderLayout.SOUTH);

        JPanel title = new JPanel();
        double[] colRatio = {0.12, 0.12, 0.12, 0.12, 0.4, 0.12};
        title.setLayout(new RatioGridLayout(1, 6, colRatio));
        title.add(new TextLabel("Order ID"));
        title.add(new TextLabel("Buyer"));
        title.add(new TextLabel("Date"));
        title.add(new TextLabel("(Total) Amount"));
        title.add(titleItem);
        title.add(new TextLabel("Ship Address"));
        title.setBorder(new LineBorder(Color.black, 3));
        add(title, BorderLayout.NORTH);

        //order data
        add(new JLabel(""), BorderLayout.CENTER);
        updateOrders(orderList);
    }

    public void updateOrders(OrderList orderList) {
        JPanel data = new JPanel();
        data.setLayout(new BoxLayout(data, BoxLayout.Y_AXIS));
        for (int i = 0; i < orderList.numOrders(); i++)
            data.add(new OrderView(this, orderList.getOrder(i)));
        remove(_layout.getLayoutComponent(BorderLayout.CENTER));
        add(data, BorderLayout.CENTER);

        _selector.reset();
        revalidate();
    }

    public void setEditMode(boolean mode) {
        _selector.setEditMode(mode);
    }

    public boolean isSelected() {
        return _selector.isSelected();
    }
}

private class OrderView extends JPanel {
    public Order _order;
    public OrderListView _orderListView;
    public ItemListView _itemListView;
    private JTextField _textField;

    public OrderView(OrderListView parent, Order order) {
        _order = order;
        _orderListView = parent;
        addMouseListener(_orderListView._selector);

        double[] colRatio = {0.12, 0.12, 0.12, 0.12, 0.4, 0.12};
        setLayout(new RatioGridLayout(1, 6, colRatio));
        setBackground(new Color(238, 238, 238));

        String[] fields = order.get();
        add(new TextLabel(" " + fields[0], JLabel.LEFT));
        add(new TextLabel(fields[1]));
        add(new TextLabel(fields[2]));
        add(new TextLabel(fields[3] + " ", JLabel.RIGHT));

        _itemListView = new ItemListView(this, order);
    }
}

```



```

        add(_itemListView);

        _textField = new JTextField(fields[4]);
        _textField.setBorder(new LineBorder(Color.black, 1));
        _textField.setEditable(false);
        add(_textField);
    }

    public void setSelected(boolean select) {
        if (select)
            setBackground(Color.white);
        else
            setBackground(new Color(238, 238, 238));
    }

    public void setEditMode(boolean mode) {
        _textField.setEditable(mode);
        if (!mode)
            _order.setAddress(_textField.getText());
    }
}

private class ItemListView extends JPanel {
    public OrderView _orderView;

    public ItemListView(OrderView parent, Order order) {
        _orderView = parent;
        setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
        for (int i = 0; ; i++) {
            String[] fields = order.getItem(i);
            if (fields == null) break;
            add(new ItemView(this, fields, i));
        }
    }
}

private class ItemView extends JPanel {
    public int _index;
    public ItemListView _itemListView;
    public JComboBox _comboBox;

    public ItemView(ItemListView parent, String[] fields, int idx) {
        _index = idx;
        _itemListView = parent;

        addMouseListener(_itemListView._orderView._orderListView._selector);
        setLayout(new GridLayout(1, 4));
        setBorder(new LineBorder(Color.black, 1));
        setBackground(new Color(238, 238, 238));

        Color c = new Color(0,0,0,0);
        add(new JLabel(fields[0], c));
        add(new JLabel(fields[1], c));
        add(new JLabel(fields[2], c));
        String[] listStatus = {"CANCELED", "PREPARING", "SHIPPED",
"DELIVERED", "RETURNED"};
        _comboBox = new JComboBox(listStatus);
        _comboBox.setSelectedItem(fields[3]);
        _comboBox.setEnabled(false);
        add(_comboBox);
    }

    public void setSelected(boolean select) {
        if (select)
            setBackground(Color.white);
    }
}

```

```

        else
            setBackground(new Color(238, 238, 238));
    }

    public void setEditMode(boolean mode) {
        _comboBox.setEnabled(mode);
        if (!mode) {
            Order order = _itemListView._orderView._order;
            order.setItemStatus(_index,
(String)_comboBox.getSelectedItem());
        }
    }

    private class TextLabel extends JLabel {
        private int _halign = JLabel.CENTER;
        private Color _color = Color.black;
        private int _thickness = 1;

        TextLabel(String text) {
            draw(text);
        }

        TextLabel(String text, int halign) {
            _halign = halign;
            draw(text);
        }

        TextLabel(String text, Color color) {
            _color = color;
            draw(text);
        }

        TextLabel(String text, int halign, Color color) {
            _halign = halign;
            _color = color;
            draw(text);
        }

        TextLabel(String text, int halign, Color color, int thickness) {
            _halign = halign;
            _color = color;
            _thickness = thickness;
            draw(text);
        }

        private void draw(String text) {
            setText(text);
            setHorizontalAlignment(_halign);
            setBorder(new LineBorder(_color, _thickness));
        }
    }

    private class SelectListener implements MouseListener {
        public OrderView _orderView;
        public ItemView _itemView;
        public boolean _editMode = false;

        public void reset() {
            if (_itemView != null) _itemView.setSelected(false);
            if (_orderView != null) _orderView.setSelected(false);
            _itemView = null;
            _orderView = null;
        }
    }

```

```

        public void setEditMode(boolean mode) {
            _editMode = mode;
            if (_itemView != null) _itemView.setEditMode(mode);
            if (_orderView != null) _orderView.setEditMode(mode);
        }

        public boolean isSelected() {
            return (_orderView != null) ? true : false;
        }

        public void mouseClicked(MouseEvent e) {
            if (_editMode) return;

            OrderView ov;
            ItemView iv;
            if (e.getSource() instanceof ItemView) {
                iv = (ItemView)e.getSource();
                ov = iv._itemListView._orderView;
            } else {
                ov = (OrderView)e.getSource();
                iv = (ItemView)ov._itemListView.getComponent(0);
            }

            if (ov != _orderView) {
                if (_orderView != null) _orderView.setSelected(false);
                ov.setSelected(true);
                _orderView = ov;
            }
            if (iv != _itemView) {
                if (_itemView != null) _itemView.setSelected(false);
                iv.setSelected(true);
                _itemView = iv;
            }
        }

        public void mousePressed(MouseEvent e) {}
        public void mouseReleased(MouseEvent e) {}
        public void mouseEntered(MouseEvent e) {}
        public void mouseExited(MouseEvent e) {}
    }

    public static void main(String[] args) {
        OrderManagement frame = new OrderManagement(_defaultOrderPath);
    }
}

```

OrderInfo.java

```

package default_package;

import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;

public class OrderInfo extends JFrame {
    private JButton _input;
    private TitleTextField _buyerName, _address, _itemId, _itemName, _itemPrice;
}

```

```

OrderInfo(OrderManagement orderManager) {
    setTitle("Input Order Information");
    setSize(800, 300);
    double[] colRatio = {0.3, 0.6, 0.1};
    setLayout(new RatioGridLayout(1, 3, colRatio));

    _buyerName = new TitleTextField("Byuer Name");
    _address = new TitleTextField("Shipping Address");
    _itemId = new TitleTextField("Item ID");
    _itemName = new TitleTextField("Item Name");
    _itemPrice = new TitleTextField("Item Price");

    JPanel inputPanel1 = new JPanel();
    inputPanel1.setLayout(new GridLayout(2, 1));
    inputPanel1.add(_buyerName);
    inputPanel1.add(_address);
    add(inputPanel1);

    JPanel inputPanel2 = new JPanel();
    inputPanel2.setLayout(new GridLayout(1, 3));
    inputPanel2.add(_itemId);
    inputPanel2.add(_itemName);
    inputPanel2.add(_itemPrice);
    add(inputPanel2);

    _input = new JButton("INPUT");
    _input.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            setVisible(false);
            orderManager.addOrder(get());
        }
    });
    add(_input);

    pack();
    setVisible(false);
}

public void showFrame() {
    _buyerName.setText("");
    _address.setText("");
    _itemId.setText("");
    _itemName.setText("");
    _itemPrice.setText("");
    setVisible(true);
}

public String get() {
    return ":: " +
        _buyerName.getText() + " :: " +
        ":: " +
        ":: " +
        _itemId.getText() + ";" +
        _itemName.getText() + ";" +
        _itemPrice.getText() + ";" +
        "PREPARING :: " +
        _address.getText();
}

private class TitleTextField extends JPanel {
    private JLabel _title = new JLabel();
    private JTextField _text = new JTextField();

    TitleTextField(String title) {

```

```
        draw(title);
    }

    TitleTextField(String title, String text) {
        _text.setText(text);
        draw(title);
    }

    private void draw(String title) {
        setLayout(new BorderLayout());
        _title.setText(title);
        _title.setBorder(new EmptyBorder(10, 5, 10, 5));
        add(_title, BorderLayout.NORTH);
        add(_text, BorderLayout.CENTER);
    }

    public String getText() {
        return _text.getText();
    }

    public void setText(String text) {
        _text.setText(text);
    }
}

}
```