**Team1**

**Project Report**

Team R.E.D

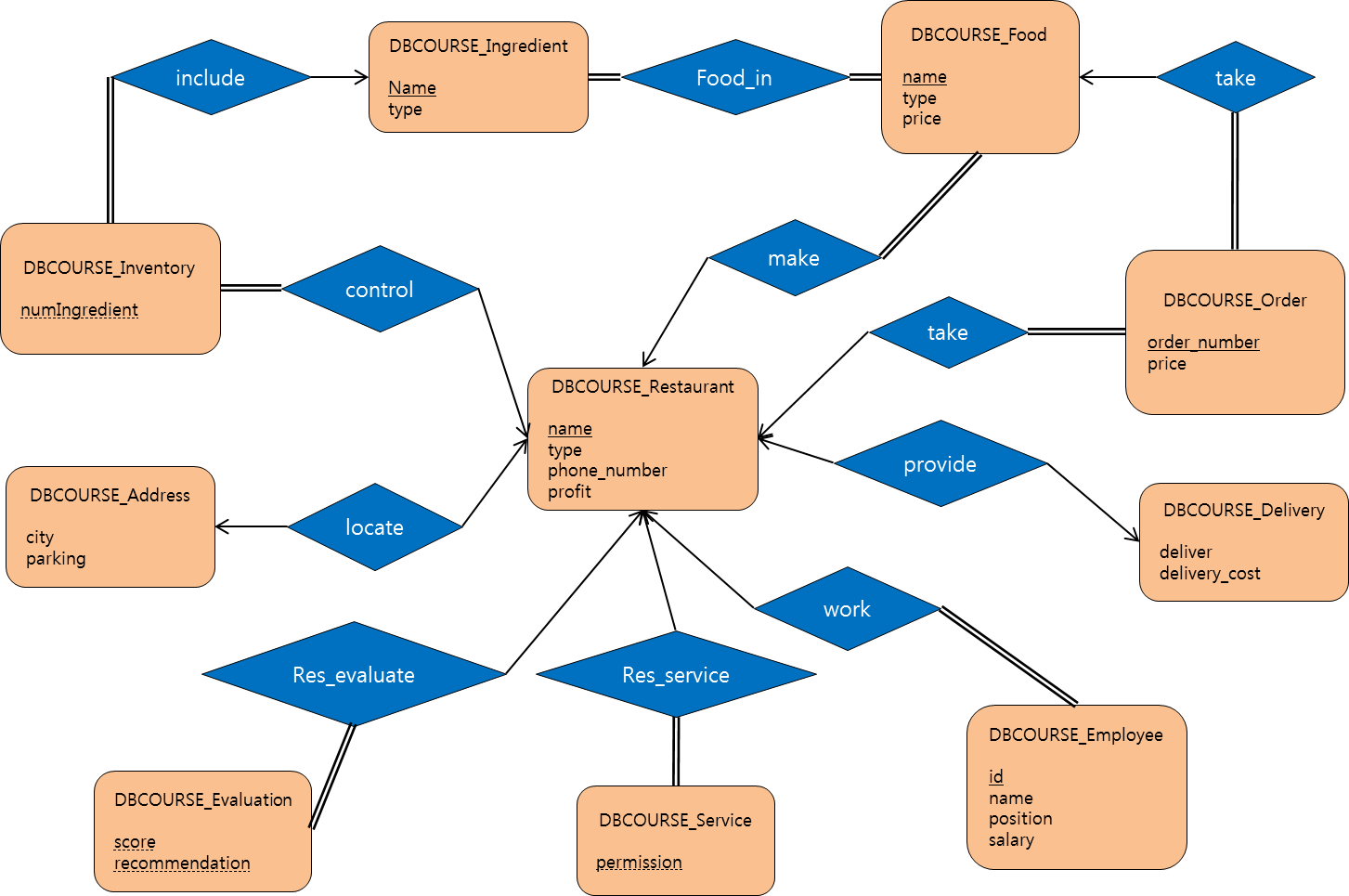
1615013 김해리

1615018 박소연

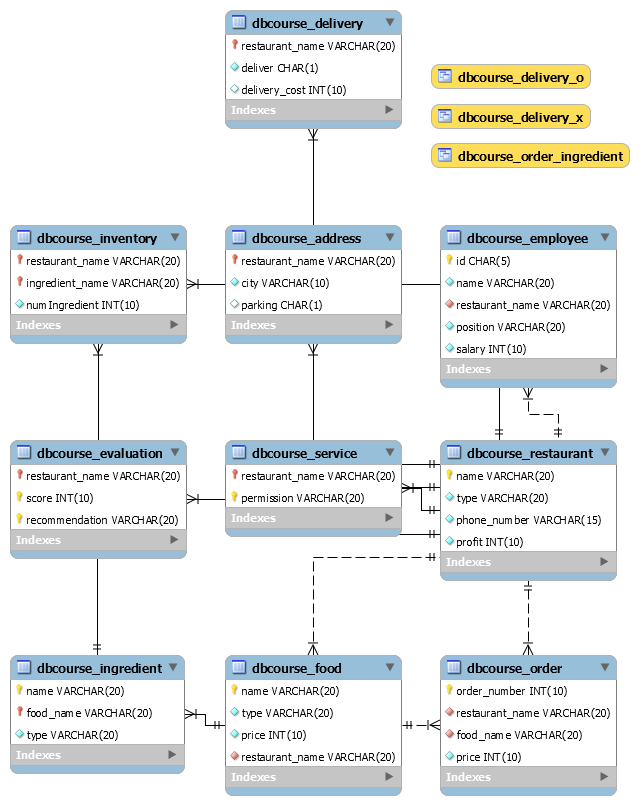
1615025 배소연

1615035 신유진

1. **ER Diagram**



1. **Database Schema Diagram**



1. **Class and Method Explanation**

## Class Test

* [java.lang.Object](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true)
  + team1.Test

public class Test

extends [Object](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true)

* + ***Nested Class Summary***

|  |  |
| --- | --- |
| **Nested Classes** | |
| **Modifier and Type** | **Class and Description** |
| static class | [Test.REDSystem](file:///C:\Users\박소연\Desktop\Test\doc\team1\Test.REDSystem.html) |

* + ***Constructor Summary***

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [Test](file:///C:\Users\박소연\Desktop\Test\doc\team1\Test.html#Test--)() |

* + ***Method Summary***

|  |  |
| --- | --- |
| **All Methods**[**Static Methods**](javascript:show(1);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| static void | [main](file:///C:\Users\박소연\Desktop\Test\doc\team1\Test.html#main-java.lang.String:A-)([**String**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html?is-external=true)[] args) |

* + - **Methods inherited from class java.lang.**[**Object**](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true)

[equals](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#equals-java.lang.Object-), [getClass](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#getClass--), [hashCode](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#hashCode--), [notify](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#notify--), [notifyAll](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#notifyAll--), [toString](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#toString--), [wait](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#wait--), [wait](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#wait-long-), [wait](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#wait-long-int-)

* + ***Constructor Detail***
    - **Test**

public Test()

* + ***Method Detail***
    - **main**

public static void main([String](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html?is-external=true)[] args)

## Class Test.REDSystem

* [java.lang.Object](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true)
  + [java.awt.Component](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true)
    - [java.awt.Container](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true)
      * [java.awt.Window](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true)
        + [java.awt.Frame](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true)

[javax.swing.JFrame](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true)

team1.Test.REDSystem

* **All Implemented Interfaces:**

[ActionListener](https://docs.oracle.com/javase/8/docs/api/java/awt/event/ActionListener.html?is-external=true), [ItemListener](https://docs.oracle.com/javase/8/docs/api/java/awt/event/ItemListener.html?is-external=true), [ImageObserver](https://docs.oracle.com/javase/8/docs/api/java/awt/image/ImageObserver.html?is-external=true), [MenuContainer](https://docs.oracle.com/javase/8/docs/api/java/awt/MenuContainer.html?is-external=true), [Serializable](https://docs.oracle.com/javase/8/docs/api/java/io/Serializable.html?is-external=true), [EventListener](https://docs.oracle.com/javase/8/docs/api/java/util/EventListener.html?is-external=true), [Accessible](https://docs.oracle.com/javase/8/docs/api/javax/accessibility/Accessible.html?is-external=true), [RootPaneContainer](https://docs.oracle.com/javase/8/docs/api/javax/swing/RootPaneContainer.html?is-external=true), [WindowConstants](https://docs.oracle.com/javase/8/docs/api/javax/swing/WindowConstants.html?is-external=true)

**Enclosing class:**

[Test](file:///C:\Users\박소연\Desktop\Test\doc\team1\Test.html)

public static class Test.REDSystem

extends [JFrame](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true)

implements [ActionListener](https://docs.oracle.com/javase/8/docs/api/java/awt/event/ActionListener.html?is-external=true), [ItemListener](https://docs.oracle.com/javase/8/docs/api/java/awt/event/ItemListener.html?is-external=true)

See Also:

[Serialized Form](file:///C:\Users\박소연\Desktop\Test\doc\serialized-form.html#team1.Test.REDSystem)

* + ***Nested Class Summary***
    - **Nested classes/interfaces inherited from class java.awt.**[**Window**](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true)

[Window.Type](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.Type.html?is-external=true)

* + - **Nested classes/interfaces inherited from class java.awt.**[**Component**](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true)

[Component.BaselineResizeBehavior](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.BaselineResizeBehavior.html?is-external=true)

* + ***Field Summary***
    - **Fields inherited from class javax.swing.**[**JFrame**](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true)

[EXIT\_ON\_CLOSE](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#EXIT_ON_CLOSE)

* + - **Fields inherited from class java.awt.**[**Frame**](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true)

[CROSSHAIR\_CURSOR](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#CROSSHAIR_CURSOR), [DEFAULT\_CURSOR](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#DEFAULT_CURSOR), [E\_RESIZE\_CURSOR](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#E_RESIZE_CURSOR), [HAND\_CURSOR](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#HAND_CURSOR), [ICONIFIED](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#ICONIFIED), [MAXIMIZED\_BOTH](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#MAXIMIZED_BOTH), [MAXIMIZED\_HORIZ](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#MAXIMIZED_HORIZ), [MAXIMIZED\_VERT](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#MAXIMIZED_VERT), [MOVE\_CURSOR](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#MOVE_CURSOR), [N\_RESIZE\_CURSOR](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#N_RESIZE_CURSOR), [NE\_RESIZE\_CURSOR](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#NE_RESIZE_CURSOR), [NORMAL](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#NORMAL), [NW\_RESIZE\_CURSOR](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#NW_RESIZE_CURSOR), [S\_RESIZE\_CURSOR](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#S_RESIZE_CURSOR), [SE\_RESIZE\_CURSOR](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#SE_RESIZE_CURSOR), [SW\_RESIZE\_CURSOR](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#SW_RESIZE_CURSOR), [TEXT\_CURSOR](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#TEXT_CURSOR), [W\_RESIZE\_CURSOR](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#W_RESIZE_CURSOR), [WAIT\_CURSOR](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#WAIT_CURSOR)

* + - **Fields inherited from class java.awt.**[**Component**](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true)

[BOTTOM\_ALIGNMENT](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#BOTTOM_ALIGNMENT), [CENTER\_ALIGNMENT](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#CENTER_ALIGNMENT), [LEFT\_ALIGNMENT](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#LEFT_ALIGNMENT), [RIGHT\_ALIGNMENT](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#RIGHT_ALIGNMENT), [TOP\_ALIGNMENT](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#TOP_ALIGNMENT)

* + - **Fields inherited from interface javax.swing.**[**WindowConstants**](https://docs.oracle.com/javase/8/docs/api/javax/swing/WindowConstants.html?is-external=true)

[DISPOSE\_ON\_CLOSE](https://docs.oracle.com/javase/8/docs/api/javax/swing/WindowConstants.html?is-external=true#DISPOSE_ON_CLOSE), [DO\_NOTHING\_ON\_CLOSE](https://docs.oracle.com/javase/8/docs/api/javax/swing/WindowConstants.html?is-external=true#DO_NOTHING_ON_CLOSE), [HIDE\_ON\_CLOSE](https://docs.oracle.com/javase/8/docs/api/javax/swing/WindowConstants.html?is-external=true#HIDE_ON_CLOSE)

* + - **Fields inherited from interface java.awt.image.**[**ImageObserver**](https://docs.oracle.com/javase/8/docs/api/java/awt/image/ImageObserver.html?is-external=true)

[ABORT](https://docs.oracle.com/javase/8/docs/api/java/awt/image/ImageObserver.html?is-external=true#ABORT), [ALLBITS](https://docs.oracle.com/javase/8/docs/api/java/awt/image/ImageObserver.html?is-external=true#ALLBITS), [ERROR](https://docs.oracle.com/javase/8/docs/api/java/awt/image/ImageObserver.html?is-external=true#ERROR), [FRAMEBITS](https://docs.oracle.com/javase/8/docs/api/java/awt/image/ImageObserver.html?is-external=true#FRAMEBITS), [HEIGHT](https://docs.oracle.com/javase/8/docs/api/java/awt/image/ImageObserver.html?is-external=true#HEIGHT), [PROPERTIES](https://docs.oracle.com/javase/8/docs/api/java/awt/image/ImageObserver.html?is-external=true#PROPERTIES), [SOMEBITS](https://docs.oracle.com/javase/8/docs/api/java/awt/image/ImageObserver.html?is-external=true#SOMEBITS), [WIDTH](https://docs.oracle.com/javase/8/docs/api/java/awt/image/ImageObserver.html?is-external=true#WIDTH)

* + ***Constructor Summary***

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [REDSystem](file:///C:\Users\박소연\Desktop\Test\doc\team1\Test.REDSystem.html#REDSystem--)() |

* + ***Method Summary***

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| void | [actionPerformed](file:///C:\Users\박소연\Desktop\Test\doc\team1\Test.REDSystem.html#actionPerformed-java.awt.event.ActionEvent-)([**ActionEvent**](https://docs.oracle.com/javase/8/docs/api/java/awt/event/ActionEvent.html?is-external=true) e) |
| void | [itemStateChanged](file:///C:\Users\박소연\Desktop\Test\doc\team1\Test.REDSystem.html#itemStateChanged-java.awt.event.ItemEvent-)([**ItemEvent**](https://docs.oracle.com/javase/8/docs/api/java/awt/event/ItemEvent.html?is-external=true) arg0) |

* + - **Methods inherited from class javax.swing.**[**JFrame**](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true)

[getAccessibleContext](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#getAccessibleContext--), [getContentPane](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#getContentPane--), [getDefaultCloseOperation](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#getDefaultCloseOperation--), [getGlassPane](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#getGlassPane--), [getGraphics](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#getGraphics--), [getJMenuBar](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#getJMenuBar--), [getLayeredPane](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#getLayeredPane--), [getRootPane](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#getRootPane--), [getTransferHandler](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#getTransferHandler--), [isDefaultLookAndFeelDecorated](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#isDefaultLookAndFeelDecorated--), [remove](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#remove-java.awt.Component-), [repaint](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#repaint-long-int-int-int-int-), [setContentPane](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#setContentPane-java.awt.Container-), [setDefaultCloseOperation](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#setDefaultCloseOperation-int-), [setDefaultLookAndFeelDecorated](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#setDefaultLookAndFeelDecorated-boolean-), [setGlassPane](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#setGlassPane-java.awt.Component-), [setIconImage](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#setIconImage-java.awt.Image-), [setJMenuBar](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#setJMenuBar-javax.swing.JMenuBar-), [setLayeredPane](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#setLayeredPane-javax.swing.JLayeredPane-), [setLayout](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#setLayout-java.awt.LayoutManager-), [setTransferHandler](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#setTransferHandler-javax.swing.TransferHandler-), [update](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html?is-external=true#update-java.awt.Graphics-)

* + - **Methods inherited from class java.awt.**[**Frame**](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true)

[addNotify](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#addNotify--), [getCursorType](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#getCursorType--), [getExtendedState](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#getExtendedState--), [getFrames](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#getFrames--), [getIconImage](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#getIconImage--), [getMaximizedBounds](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#getMaximizedBounds--), [getMenuBar](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#getMenuBar--), [getState](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#getState--), [getTitle](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#getTitle--), [isResizable](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#isResizable--), [isUndecorated](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#isUndecorated--), [remove](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#remove-java.awt.MenuComponent-), [removeNotify](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#removeNotify--), [setBackground](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#setBackground-java.awt.Color-), [setCursor](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#setCursor-int-), [setExtendedState](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#setExtendedState-int-), [setMaximizedBounds](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#setMaximizedBounds-java.awt.Rectangle-), [setMenuBar](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#setMenuBar-java.awt.MenuBar-), [setOpacity](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#setOpacity-float-), [setResizable](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#setResizable-boolean-), [setShape](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#setShape-java.awt.Shape-), [setState](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#setState-int-), [setTitle](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#setTitle-java.lang.String-), [setUndecorated](https://docs.oracle.com/javase/8/docs/api/java/awt/Frame.html?is-external=true#setUndecorated-boolean-)

* + - **Methods inherited from class java.awt.**[**Window**](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true)

[addPropertyChangeListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#addPropertyChangeListener-java.beans.PropertyChangeListener-), [addPropertyChangeListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#addPropertyChangeListener-java.lang.String-java.beans.PropertyChangeListener-), [addWindowFocusListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#addWindowFocusListener-java.awt.event.WindowFocusListener-), [addWindowListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#addWindowListener-java.awt.event.WindowListener-), [addWindowStateListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#addWindowStateListener-java.awt.event.WindowStateListener-), [applyResourceBundle](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#applyResourceBundle-java.util.ResourceBundle-), [applyResourceBundle](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#applyResourceBundle-java.lang.String-), [createBufferStrategy](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#createBufferStrategy-int-), [createBufferStrategy](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#createBufferStrategy-int-java.awt.BufferCapabilities-), [dispose](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#dispose--), [getBackground](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getBackground--), [getBufferStrategy](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getBufferStrategy--), [getFocusableWindowState](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getFocusableWindowState--), [getFocusCycleRootAncestor](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getFocusCycleRootAncestor--), [getFocusOwner](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getFocusOwner--), [getFocusTraversalKeys](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getFocusTraversalKeys-int-), [getIconImages](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getIconImages--), [getInputContext](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getInputContext--), [getListeners](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getListeners-java.lang.Class-), [getLocale](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getLocale--), [getModalExclusionType](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getModalExclusionType--), [getMostRecentFocusOwner](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getMostRecentFocusOwner--), [getOpacity](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getOpacity--), [getOwnedWindows](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getOwnedWindows--), [getOwner](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getOwner--), [getOwnerlessWindows](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getOwnerlessWindows--), [getShape](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getShape--), [getToolkit](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getToolkit--), [getType](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getType--), [getWarningString](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getWarningString--), [getWindowFocusListeners](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getWindowFocusListeners--), [getWindowListeners](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getWindowListeners--), [getWindows](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getWindows--), [getWindowStateListeners](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#getWindowStateListeners--), [hide](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#hide--), [isActive](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#isActive--), [isAlwaysOnTop](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#isAlwaysOnTop--), [isAlwaysOnTopSupported](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#isAlwaysOnTopSupported--), [isAutoRequestFocus](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#isAutoRequestFocus--), [isFocusableWindow](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#isFocusableWindow--), [isFocusCycleRoot](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#isFocusCycleRoot--), [isFocused](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#isFocused--), [isLocationByPlatform](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#isLocationByPlatform--), [isOpaque](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#isOpaque--), [isShowing](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#isShowing--), [isValidateRoot](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#isValidateRoot--), [pack](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#pack--), [paint](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#paint-java.awt.Graphics-), [postEvent](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#postEvent-java.awt.Event-), [removeWindowFocusListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#removeWindowFocusListener-java.awt.event.WindowFocusListener-), [removeWindowListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#removeWindowListener-java.awt.event.WindowListener-), [removeWindowStateListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#removeWindowStateListener-java.awt.event.WindowStateListener-), [reshape](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#reshape-int-int-int-int-), [setAlwaysOnTop](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setAlwaysOnTop-boolean-), [setAutoRequestFocus](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setAutoRequestFocus-boolean-), [setBounds](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setBounds-int-int-int-int-), [setBounds](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setBounds-java.awt.Rectangle-), [setCursor](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setCursor-java.awt.Cursor-), [setFocusableWindowState](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setFocusableWindowState-boolean-), [setFocusCycleRoot](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setFocusCycleRoot-boolean-), [setIconImages](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setIconImages-java.util.List-), [setLocation](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setLocation-int-int-), [setLocation](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setLocation-java.awt.Point-), [setLocationByPlatform](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setLocationByPlatform-boolean-), [setLocationRelativeTo](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setLocationRelativeTo-java.awt.Component-), [setMinimumSize](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setMinimumSize-java.awt.Dimension-), [setModalExclusionType](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setModalExclusionType-java.awt.Dialog.ModalExclusionType-), [setSize](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setSize-java.awt.Dimension-), [setSize](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setSize-int-int-), [setType](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setType-java.awt.Window.Type-), [setVisible](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#setVisible-boolean-), [show](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#show--), [toBack](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#toBack--), [toFront](https://docs.oracle.com/javase/8/docs/api/java/awt/Window.html?is-external=true#toFront--)

* + - **Methods inherited from class java.awt.**[**Container**](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true)

[add](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#add-java.awt.Component-), [add](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#add-java.awt.Component-int-), [add](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#add-java.awt.Component-java.lang.Object-), [add](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#add-java.awt.Component-java.lang.Object-int-), [add](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#add-java.lang.String-java.awt.Component-), [addContainerListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#addContainerListener-java.awt.event.ContainerListener-), [applyComponentOrientation](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#applyComponentOrientation-java.awt.ComponentOrientation-), [areFocusTraversalKeysSet](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#areFocusTraversalKeysSet-int-), [countComponents](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#countComponents--), [deliverEvent](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#deliverEvent-java.awt.Event-), [doLayout](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#doLayout--), [findComponentAt](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#findComponentAt-int-int-), [findComponentAt](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#findComponentAt-java.awt.Point-), [getAlignmentX](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#getAlignmentX--), [getAlignmentY](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#getAlignmentY--), [getComponent](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#getComponent-int-), [getComponentAt](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#getComponentAt-int-int-), [getComponentAt](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#getComponentAt-java.awt.Point-), [getComponentCount](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#getComponentCount--), [getComponents](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#getComponents--), [getComponentZOrder](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#getComponentZOrder-java.awt.Component-), [getContainerListeners](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#getContainerListeners--), [getFocusTraversalPolicy](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#getFocusTraversalPolicy--), [getInsets](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#getInsets--), [getLayout](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#getLayout--), [getMaximumSize](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#getMaximumSize--), [getMinimumSize](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#getMinimumSize--), [getMousePosition](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#getMousePosition-boolean-), [getPreferredSize](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#getPreferredSize--), [insets](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#insets--), [invalidate](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#invalidate--), [isAncestorOf](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#isAncestorOf-java.awt.Component-), [isFocusCycleRoot](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#isFocusCycleRoot-java.awt.Container-), [isFocusTraversalPolicyProvider](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#isFocusTraversalPolicyProvider--), [isFocusTraversalPolicySet](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#isFocusTraversalPolicySet--), [layout](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#layout--), [list](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#list-java.io.PrintStream-int-), [list](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#list-java.io.PrintWriter-int-), [locate](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#locate-int-int-), [minimumSize](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#minimumSize--), [paintComponents](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#paintComponents-java.awt.Graphics-), [preferredSize](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#preferredSize--), [print](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#print-java.awt.Graphics-), [printComponents](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#printComponents-java.awt.Graphics-), [remove](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#remove-int-), [removeAll](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#removeAll--), [removeContainerListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#removeContainerListener-java.awt.event.ContainerListener-), [setComponentZOrder](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#setComponentZOrder-java.awt.Component-int-), [setFocusTraversalKeys](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#setFocusTraversalKeys-int-java.util.Set-), [setFocusTraversalPolicy](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#setFocusTraversalPolicy-java.awt.FocusTraversalPolicy-), [setFocusTraversalPolicyProvider](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#setFocusTraversalPolicyProvider-boolean-), [setFont](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#setFont-java.awt.Font-), [transferFocusDownCycle](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#transferFocusDownCycle--), [validate](https://docs.oracle.com/javase/8/docs/api/java/awt/Container.html?is-external=true#validate--)

* + - **Methods inherited from class java.awt.**[**Component**](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true)

[action](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#action-java.awt.Event-java.lang.Object-), [add](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#add-java.awt.PopupMenu-), [addComponentListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#addComponentListener-java.awt.event.ComponentListener-), [addFocusListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#addFocusListener-java.awt.event.FocusListener-), [addHierarchyBoundsListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#addHierarchyBoundsListener-java.awt.event.HierarchyBoundsListener-), [addHierarchyListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#addHierarchyListener-java.awt.event.HierarchyListener-), [addInputMethodListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#addInputMethodListener-java.awt.event.InputMethodListener-), [addKeyListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#addKeyListener-java.awt.event.KeyListener-), [addMouseListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#addMouseListener-java.awt.event.MouseListener-), [addMouseMotionListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#addMouseMotionListener-java.awt.event.MouseMotionListener-), [addMouseWheelListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#addMouseWheelListener-java.awt.event.MouseWheelListener-), [bounds](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#bounds--), [checkImage](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#checkImage-java.awt.Image-java.awt.image.ImageObserver-), [checkImage](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#checkImage-java.awt.Image-int-int-java.awt.image.ImageObserver-), [contains](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#contains-int-int-), [contains](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#contains-java.awt.Point-), [createImage](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#createImage-java.awt.image.ImageProducer-), [createImage](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#createImage-int-int-), [createVolatileImage](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#createVolatileImage-int-int-), [createVolatileImage](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#createVolatileImage-int-int-java.awt.ImageCapabilities-), [disable](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#disable--), [dispatchEvent](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#dispatchEvent-java.awt.AWTEvent-), [enable](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#enable--), [enable](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#enable-boolean-), [enableInputMethods](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#enableInputMethods-boolean-), [firePropertyChange](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#firePropertyChange-java.lang.String-byte-byte-), [firePropertyChange](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#firePropertyChange-java.lang.String-char-char-), [firePropertyChange](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#firePropertyChange-java.lang.String-double-double-), [firePropertyChange](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#firePropertyChange-java.lang.String-float-float-), [firePropertyChange](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#firePropertyChange-java.lang.String-long-long-), [firePropertyChange](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#firePropertyChange-java.lang.String-short-short-), [getBaseline](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getBaseline-int-int-), [getBaselineResizeBehavior](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getBaselineResizeBehavior--), [getBounds](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getBounds--), [getBounds](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getBounds-java.awt.Rectangle-), [getColorModel](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getColorModel--), [getComponentListeners](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getComponentListeners--), [getComponentOrientation](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getComponentOrientation--), [getCursor](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getCursor--), [getDropTarget](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getDropTarget--), [getFocusListeners](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getFocusListeners--), [getFocusTraversalKeysEnabled](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getFocusTraversalKeysEnabled--), [getFont](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getFont--), [getFontMetrics](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getFontMetrics-java.awt.Font-), [getForeground](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getForeground--), [getGraphicsConfiguration](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getGraphicsConfiguration--), [getHeight](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getHeight--), [getHierarchyBoundsListeners](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getHierarchyBoundsListeners--), [getHierarchyListeners](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getHierarchyListeners--), [getIgnoreRepaint](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getIgnoreRepaint--), [getInputMethodListeners](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getInputMethodListeners--), [getInputMethodRequests](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getInputMethodRequests--), [getKeyListeners](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getKeyListeners--), [getLocation](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getLocation--), [getLocation](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getLocation-java.awt.Point-), [getLocationOnScreen](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getLocationOnScreen--), [getMouseListeners](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getMouseListeners--), [getMouseMotionListeners](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getMouseMotionListeners--), [getMousePosition](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getMousePosition--), [getMouseWheelListeners](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getMouseWheelListeners--), [getName](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getName--), [getParent](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getParent--), [getPeer](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getPeer--), [getPropertyChangeListeners](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getPropertyChangeListeners--), [getPropertyChangeListeners](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getPropertyChangeListeners-java.lang.String-), [getSize](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getSize--), [getSize](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getSize-java.awt.Dimension-), [getTreeLock](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getTreeLock--), [getWidth](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getWidth--), [getX](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getX--), [getY](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#getY--), [gotFocus](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#gotFocus-java.awt.Event-java.lang.Object-), [handleEvent](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#handleEvent-java.awt.Event-), [hasFocus](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#hasFocus--), [imageUpdate](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#imageUpdate-java.awt.Image-int-int-int-int-int-), [inside](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#inside-int-int-), [isBackgroundSet](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#isBackgroundSet--), [isCursorSet](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#isCursorSet--), [isDisplayable](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#isDisplayable--), [isDoubleBuffered](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#isDoubleBuffered--), [isEnabled](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#isEnabled--), [isFocusable](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#isFocusable--), [isFocusOwner](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#isFocusOwner--), [isFocusTraversable](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#isFocusTraversable--), [isFontSet](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#isFontSet--), [isForegroundSet](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#isForegroundSet--), [isLightweight](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#isLightweight--), [isMaximumSizeSet](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#isMaximumSizeSet--), [isMinimumSizeSet](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#isMinimumSizeSet--), [isPreferredSizeSet](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#isPreferredSizeSet--), [isValid](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#isValid--), [isVisible](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#isVisible--), [keyDown](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#keyDown-java.awt.Event-int-), [keyUp](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#keyUp-java.awt.Event-int-), [list](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#list--), [list](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#list-java.io.PrintStream-), [list](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#list-java.io.PrintWriter-), [location](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#location--), [lostFocus](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#lostFocus-java.awt.Event-java.lang.Object-), [mouseDown](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#mouseDown-java.awt.Event-int-int-), [mouseDrag](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#mouseDrag-java.awt.Event-int-int-), [mouseEnter](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#mouseEnter-java.awt.Event-int-int-), [mouseExit](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#mouseExit-java.awt.Event-int-int-), [mouseMove](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#mouseMove-java.awt.Event-int-int-), [mouseUp](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#mouseUp-java.awt.Event-int-int-), [move](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#move-int-int-), [nextFocus](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#nextFocus--), [paintAll](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#paintAll-java.awt.Graphics-), [prepareImage](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#prepareImage-java.awt.Image-java.awt.image.ImageObserver-), [prepareImage](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#prepareImage-java.awt.Image-int-int-java.awt.image.ImageObserver-), [printAll](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#printAll-java.awt.Graphics-), [removeComponentListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#removeComponentListener-java.awt.event.ComponentListener-), [removeFocusListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#removeFocusListener-java.awt.event.FocusListener-), [removeHierarchyBoundsListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#removeHierarchyBoundsListener-java.awt.event.HierarchyBoundsListener-), [removeHierarchyListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#removeHierarchyListener-java.awt.event.HierarchyListener-), [removeInputMethodListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#removeInputMethodListener-java.awt.event.InputMethodListener-), [removeKeyListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#removeKeyListener-java.awt.event.KeyListener-), [removeMouseListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#removeMouseListener-java.awt.event.MouseListener-), [removeMouseMotionListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#removeMouseMotionListener-java.awt.event.MouseMotionListener-), [removeMouseWheelListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#removeMouseWheelListener-java.awt.event.MouseWheelListener-), [removePropertyChangeListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#removePropertyChangeListener-java.beans.PropertyChangeListener-), [removePropertyChangeListener](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#removePropertyChangeListener-java.lang.String-java.beans.PropertyChangeListener-), [repaint](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#repaint--), [repaint](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#repaint-int-int-int-int-), [repaint](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#repaint-long-), [requestFocus](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#requestFocus--), [requestFocusInWindow](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#requestFocusInWindow--), [resize](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#resize-java.awt.Dimension-), [resize](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#resize-int-int-), [revalidate](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#revalidate--), [setComponentOrientation](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#setComponentOrientation-java.awt.ComponentOrientation-), [setDropTarget](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#setDropTarget-java.awt.dnd.DropTarget-), [setEnabled](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#setEnabled-boolean-), [setFocusable](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#setFocusable-boolean-), [setFocusTraversalKeysEnabled](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#setFocusTraversalKeysEnabled-boolean-), [setForeground](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#setForeground-java.awt.Color-), [setIgnoreRepaint](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#setIgnoreRepaint-boolean-), [setLocale](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#setLocale-java.util.Locale-), [setMaximumSize](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#setMaximumSize-java.awt.Dimension-), [setName](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#setName-java.lang.String-), [setPreferredSize](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#setPreferredSize-java.awt.Dimension-), [show](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#show-boolean-), [size](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#size--), [toString](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#toString--), [transferFocus](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#transferFocus--), [transferFocusBackward](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#transferFocusBackward--), [transferFocusUpCycle](https://docs.oracle.com/javase/8/docs/api/java/awt/Component.html?is-external=true#transferFocusUpCycle--)

* + - **Methods inherited from class java.lang.**[**Object**](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true)

[equals](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#equals-java.lang.Object-), [getClass](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#getClass--), [hashCode](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#hashCode--), [notify](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#notify--), [notifyAll](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#notifyAll--), [wait](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#wait--), [wait](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#wait-long-), [wait](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true#wait-long-int-)

* + - **Methods inherited from interface java.awt.**[**MenuContainer**](https://docs.oracle.com/javase/8/docs/api/java/awt/MenuContainer.html?is-external=true)

[getFont](https://docs.oracle.com/javase/8/docs/api/java/awt/MenuContainer.html?is-external=true#getFont--), [postEvent](https://docs.oracle.com/javase/8/docs/api/java/awt/MenuContainer.html?is-external=true#postEvent-java.awt.Event-)

* + ***Constructor Detail***
    - **REDSystem**

public REDSystem()

* + ***Method Detail***
    - **actionPerformed**

public void actionPerformed([ActionEvent](https://docs.oracle.com/javase/8/docs/api/java/awt/event/ActionEvent.html?is-external=true) e)

Specified by:

[actionPerformed](https://docs.oracle.com/javase/8/docs/api/java/awt/event/ActionListener.html?is-external=true#actionPerformed-java.awt.event.ActionEvent-) in interface [ActionListener](https://docs.oracle.com/javase/8/docs/api/java/awt/event/ActionListener.html?is-external=true)

* + - **itemStateChanged**

public void itemStateChanged([ItemEvent](https://docs.oracle.com/javase/8/docs/api/java/awt/event/ItemEvent.html?is-external=true) arg0)

Specified by:

[itemStateChanged](https://docs.oracle.com/javase/8/docs/api/java/awt/event/ItemListener.html?is-external=true#itemStateChanged-java.awt.event.ItemEvent-) in interface [ItemListener](https://docs.oracle.com/javase/8/docs/api/java/awt/event/ItemListener.html?is-external=true)

1. **Main class name and How to run, Connection configuration**

Main class name : Test

Connection configuration :

|  |  |
| --- | --- |
| //jdbc driver 이용      static final String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";      //database는 team1      static final String DB\_URL = "jdbc:mysql://localhost:3306/team1?useSSL=false&serverTimezone=UTC";      //user는 team1      static final String USER = "team1";      // password는 team1      static final String PASS = "team1";      //커넥션 설정을 위한 초기화      static  Connection conn = null;      //스테이트먼트 실행을 위한 초기화      static   Statement stmt = null;      static   Statement stmt2 = null;        public static void main(String[] args) {          try {              //Register JDBC driver              Class.forName(JDBC\_DRIVER);              //Open a connection              conn = DriverManager.getConnection(DB\_URL, USER, PASS);              //Execute a query              stmt = conn.createStatement();              stmt2 = conn.createStatement();          }catch(Exception e) {              e.printStackTrace();          }          //frame 생성          REDSystem my = new REDSystem();      }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

1. **17 requirements**

* Tables & columns:

1. DBCOURSE\_Restaurant – name, type, phone\_number, profit
2. DBCOURSE\_Ingredient – name, food\_name, type
3. DBCOURSE\_Food – name, type, price, restaurant\_name
4. DBCOURSE\_Employee – id, name, restaurant\_name, position, salary
5. DBCOURSE\_Evaluation – restaurant\_name, score, recommendation
6. DBCOURSE\_Address – restaurant\_name, city, parking
7. DBCOURSE\_Delivery – restaurant\_name, deliver, delivery\_cost
8. DBCOURSE\_Service – restaurant\_name, permission
9. DBCOURSE\_Order – order\_number, restaurant\_name, food\_name, price
10. DBCOURSE\_Inventory – restaurant\_name, ingredient\_name, numIngredient

* 10 tables 34 columns
* At least 50 records inserted:

1. DBCOURSE\_Restaurant – 11 records
2. DBCOURSE\_Ingredient – 59 records
3. DBCOURSE\_Food – 33 records
4. DBCOURSE\_Employee – 29 records
5. DBCOURSE\_Evaluation – 31 records
6. DBCOURSE\_Address – 11 records
7. DBCOURSE\_Delivery – 11 records
8. DBCOURSE\_Service – 7 records
9. DBCOURSE\_Order – 0 records
10. DBCOURSE\_Inventory – 43 records

* 235 records
* Primary key, foreign key, not null constraints

|  |  |
| --- | --- |
| create table DBCOURSE\_Food (      name varchar(20) primary key,      type varchar(20) not null,      price integer(10) not null,      restaurant\_name varchar(20) not null,      foreign key(restaurant\_name) references DBCOURSE\_Restaurant(name)); | [cs](http://colorscripter.com/info#e) |

* At least 2 views

|  |  |
| --- | --- |
| create view DBCOURSE\_Delivery\_O as      select DBCOURSE\_Restaurant.name as restaurant\_name, city,             DBCOURSE\_Food.type as type, DBCOURSE\_Food.name as food\_name, price      from (DBCOURSE\_Delivery natural join DBCOURSE\_Food,DBCOURSE\_Restaurant)            natural join DBCOURSE\_Address      where DBCOURSE\_Food.restaurant\_name = DBCOURSE\_Restaurant.name and deliver = 'o'      group by DBCOURSE\_Food.name;    create view DBCOURSE\_Delivery\_X as      select DBCOURSE\_Restaurant.name as restaurant\_name, city,             DBCOURSE\_Food.type as type, DBCOURSE\_Food.name as food\_name, price      from (DBCOURSE\_Delivery natural join DBCOURSE\_Food,DBCOURSE\_Restaurant)            natural join DBCOURSE\_Address      where DBCOURSE\_Food.restaurant\_name = DBCOURSE\_Restaurant.name and deliver = 'x'      group by DBCOURSE\_Food.name;    create view DBCOURSE\_Order\_Ingredient as      select order\_number, DBCOURSE\_Order.restaurant\_name,             DBCOURSE\_Order.food\_name, DBCOURSE\_Ingredient.name as ingredient      from DBCOURSE\_Food, DBCOURSE\_Order, DBCOURSE\_Ingredient      where DBCOURSE\_Food.restaurant\_name = DBCOURSE\_Order.restaurant\_name      and DBCOURSE\_Food.name = DBCOURSE\_Order.food\_name      and DBCOURSE\_Food.name = DBCOURSE\_Ingredient.food\_name; | [cs](http://colorscripter.com/info#e) |

* 3 views
* At least 5 indexes

|  |  |
| --- | --- |
| create index restaurant\_name\_index on dbcourse\_restaurant(name);  create index restaurant\_city\_index on dbcourse\_address(city);  create index restaurant\_food\_index on dbcourse\_food(name);  create index food\_price\_index on dbcourse\_food(price);  create index employee\_id\_index on dbcourse\_employee(id);  create index ingredient\_index on dbcourse\_ingredient(name); | [cs](http://colorscripter.com/info#e) |

* All tables and view names should be prefixed with “DBCOURSE\_”

Table names : DBCOURSE\_Restaurant, DBCOURSE\_Ingredient, DBCOURSE\_Food, DBCOURSE\_Employee, DBCOURSE\_Evaluation, DBCOURSE\_Address, DBCOURSE\_Delivery, DBCOURSE\_Service, DBCOURSE\_Order, DBCOURSE\_Inventory

View names : DBCOURSE\_Delivery\_O, DBCOURSE\_Delivery\_X, DBCOURSE\_Order\_Ingredient

* Should use transactions

|  |  |
| --- | --- |
| try {      //setAutoCommit()함수에 false 인자를 주어 transaction을 실행한다      conn.setAutoCommit(false);      //음식의 이름이 foodname일때 재료를 구해서 numIngredient가 0인 재료의 개수를 센다      String S = "select count(ingredient\_name) " +              "from (select DBCOURSE\_Ingredient.food\_name, DBCOURSE\_Inventory.ingredient\_name, DBCOURSE\_Inventory.numIngredient " +              "from DBCOURSE\_Inventory, DBCOURSE\_Ingredient, DBCOURSE\_Food " +              "where (DBCOURSE\_Ingredient.food\_name = DBCOURSE\_Food.name) " +              "and (DBCOURSE\_Inventory.ingredient\_name = DBCOURSE\_Ingredient.name) " +              "and(DBCOURSE\_Inventory.restaurant\_name = DBCOURSE\_Food.restaurant\_name)) as A " +              "where numIngredient = 0 and food\_name = '" + foodname + "'";      //ingredient가 남아 있는지 확인하기 위한 count(ingredient) tuple 받아오기      ResultSet rs = stmt.executeQuery(S);      int count = 0;      //count 변수를 numIngredient가 0인 재료의 개수로 설정      if(rs.next()) count = rs.getInt(1);      if(count == 0) { //재료가 부족하지 않으면          try {              /\*\* 박소연\*/              //DBCOURSE\_Order에 insert를 할 때 preparedstatement 이용              PreparedStatement pStmt=conn.prepareStatement("insert into DBCOURSE\_Order values(?,?,?,?)");              rs = stmt.executeQuery("select max(order\_number) from DBCOURSE\_Order");  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |  |
| --- | --- | --- |
| /\*\* 배소연\*/              //주문번호는 max(order\_number)+1로 자동 셋팅              if(rs.next()) count = rs.getInt(1);              count++;                /\*\* 박소연\*/              //delivery인 경우              if(flag==0){                  pStmt.setInt(1, count);                  pStmt.setString(2, res\_name);                  pStmt.setString(3, foodname);                  //음식가격에 배달료를 더하여 음식가격을 설정                  pStmt.setInt(4, (Integer.valueOf(foodprice)+Integer.valueOf(deliverycost)));                  pStmt.executeUpdate();              }              //No delivery인 경우              else{                  pStmt.setInt(1, count);                  pStmt.setString(2, res\_name);                  pStmt.setString(3, foodname);                  pStmt.setInt(4, (Integer.valueOf(foodprice)));                  pStmt.executeUpdate();              }              /\*\* 배소연\*/              //방금 들어간 주문의 재료를 가져온다              rs = stmt.executeQuery("select ingredient " +                      "from DBCOURSE\_Order\_Ingredient " +                      "where order\_number >= all(select order\_number from DBCOURSE\_Order\_Ingredient)");              String ingr = "";              while(rs.next()) { //방금 들어간 주문의 레스토랑에서 재료를 하나씩 빼서 요리한다.                  ingr = rs.getString(1);                  stmt2.executeUpdate("update DBCOURSE\_Inventory " +                          "set numIngredient = numIngredient - 1 " +                          "where DBCOURSE\_Inventory.restaurant\_name = '" + res\_name +                          "' and DBCOURSE\_Inventory.ingredient\_name = '" + ingr + "' ");              }          } catch (SQLException e1) {              e1.printStackTrace();          }      }      else { //재료가 부족하면 X와 noingredient label을 보여준다          complete.setVisible(false);          x.setVisible(true);          noingredient.setVisible(true);      }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | | [cs](http://colorscripter.com/info#e) |
| //수행작업이 잘 이뤄졌을 때 변동사항을 적용한다      conn.commit();  } catch (SQLException e1) {      e1.printStackTrace(); //예외 발생 원인 출력      if(conn != null) {          try {              //SQLException 발생 시 rollback으로 변동사항을 없는 것으로 설정              conn.rollback();          } catch (SQLException e2) {              e2.printStackTrace();          }      }  } finally {      try { //다시 setAutoCommit을 true로 설정          conn.setAutoCommit(true);      } catch (SQLException e1) {          e1.printStackTrace();      }  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

//delete와 insert가 같이 일어나야 하기 때문에 transaction

try{

    //setAutoCommit()함수에 false 인자를 주어 transaction을 실행한다

    conn.setAutoCommit(false);

    //dbcourse\_seriesName 테이블에서 사용자가 입력한 시리즈 삭제

    stmt.executeUpdate("delete from DBCOURSE\_Employee where id='"+emp\_ID+"'");

    PreparedStatement pStmt=conn.prepareStatement("insert into DBCOURSE\_Employee values(?,?,?,?,?)");

    pStmt.setString(1, emp\_ID);

    pStmt.setString(2, newname);

    pStmt.setString(3, newresname);

    pStmt.setString(4, newposition);

    pStmt.setInt(5, newsalary);

    pStmt.executeUpdate();

    //수행작업이 잘 이뤄졌을 때 변동사항을 적용한다

    conn.commit();

} catch(SQLException se) {

    se.printStackTrace();

    if(conn != null) {

        try {

            //SQLException 발생 시 rollback으로 변동사항을 없는 것으로 설정

            conn.rollback();

        } catch (SQLException e1) {

            e1.printStackTrace();

        }

    }

} finally {

    try { //다시 setAutoCommit을 true로 설정

        conn.setAutoCommit(true);

    } catch (SQLException e1) {

        e1.printStackTrace();

    }

}

[*Colored by Color Scripter*](http://colorscripter.com/info#e)

|  |  |
| --- | --- |
| //order 처리와 profit update는 항상 같이 이뤄져야 하기 때문에 transaction  try {      //setAutoCommit()함수에 false 인자를 주어 transaction을 실행한다      conn.setAutoCommit(false);      //order number가 가장 작은 tuple을 고른다      ResultSet rs = stmt.executeQuery("select restaurant\_name, price from DBCOURSE\_Order " +              "where order\_number <= all(select order\_number from DBCOURSE\_Order)");      if(rs.next()) {          res = rs.getString(1);          price = rs.getInt(2);      }      //가져온 restaurant name과 price로 profit을 update한다      stmt.executeUpdate("update DBCOURSE\_Restaurant " +              "set profit = profit + " + price +              " where name = '" + res + "'");      //처리가 끝난 order를 order table에서 delete한다      stmt2.executeUpdate("delete from DBCOURSE\_Order " +              "where order\_number <= all(select order\_number from (select order\_number from DBCOURSE\_Order) as A)");        //남은 order table의 tuple을 가져온다      rs = stmt.executeQuery("select \* from DBCOURSE\_Order");      while(rs.next()) {          Object data[] = {rs.getInt(1), rs.getString(2), rs.getString(3), rs.getInt(4)};          model.addRow(data);      }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //update된 restaurant 별 현재의 profit을 가져온다      rs = stmt.executeQuery("select name as restaurant\_name, profit from DBCOURSE\_Restaurant");      while(rs.next()) {          Object data[] = {rs.getString(1), rs.getInt(2)};          model2.addRow(data);      }      //수행작업이 잘 이뤄졌을 때 변동사항을 적용한다      conn.commit();  } catch (SQLException e1) {      e1.printStackTrace();      if(conn != null) {          try {              //SQLException 발생 시 rollback으로 변동사항을 없는 것으로 설정              conn.rollback();          } catch (SQLException e2) {              e2.printStackTrace();          }      }  } finally {      try { //다시 setAutoCommit을 true로 설정          conn.setAutoCommit(true);      } catch (SQLException e1) {          e1.printStackTrace();      }  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

* Should include queries that use index (query containing indexed column)

|  |  |
| --- | --- |
| //레스토랑의 delivery cost를 받아온다  String S = "select delivery\_cost from DBCOURSE\_Delivery where restaurant\_name='"+res\_name+"'";  ResultSet rs = stmt.executeQuery(S);  while(rs.next()){      deliverycost=String.valueOf(rs.getInt(1));      delivery\_cost.setText(deliverycost+" won");  }  delivery\_cost\_label.setText("Delivery Cost: ");  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //체크한 도시를 추가하는 for문  for(int i=0; i<check.length; i++) {      if(check[i].getState() == true) {          count++;          if(count == 1) S += " where city = '" + check[i].getLabel() + "'";          else S += " or city = '" + check[i].getLabel() + "'";      }  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| String temp = "";  ResultSet rs = stmt.executeQuery(S2);  while(rs.next()) { //선택된 모든 ingredient를 하나씩 빼는 while문      count++;      if(count == 1) {//처음 체크된 경우에만 where문 추가          /\*\* sql - 김해리, 신유진\*/          if(flag == 0) S += " where food\_name not in " +                  "(select DBCOURSE\_Delivery\_O.food\_name from DBCOURSE\_Delivery\_O, DBCOURSE\_Ingredient " +                  "where DBCOURSE\_Delivery\_O.food\_name = DBCOURSE\_Ingredient.food\_name " +                  "and (DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";          else S += S += " where food\_name not in " +                  "(select DBCOURSE\_Delivery\_X.food\_name from DBCOURSE\_Delivery\_X, DBCOURSE\_Ingredient " +                  "where DBCOURSE\_Delivery\_X.food\_name = DBCOURSE\_Ingredient.food\_name " +                  "and (DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";      }      else S += " or DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";  }  if(flag == 0) S += ") group by DBCOURSE\_Delivery\_O.food\_name)";  else S += ") group by DBCOURSE\_Delivery\_X.food\_name)";  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //price 기준을 선택했을 때  else if(price\_radio.isSelected()) {      //price 조건을 제대로 선택한 경우 success label을 보여준다      select.setVisible(false);      try\_again.setVisible(false);      success.setVisible(true);      //delivery인 경우      if(flag == 0) S = "select \* from DBCOURSE\_Delivery\_O";      //No delivery인 경우      else S = "select \* from DBCOURSE\_Delivery\_X";      //선택한 조건에 따라 where문 추가      if(r5000.isSelected()) S += " where price < 5000";      else if(r10000.isSelected()) S += " where price < 10000";      else if(r15000.isSelected()) S += " where price < 15000";      else if(r20000.isSelected()) S += " where price < 20000";      else if(r25000.isSelected()) S += " where price < 25000";      else if(r30000.isSelected()) S += " where price < 30000"; | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //employee id로 찾은 employee 정보를 DBCOURSE\_Employee에서 제거한다  try{      stmt.executeUpdate("delete from DBCOURSE\_Employee where id='"+emp\_ID+"'");  } catch(SQLException se) {      se.printStackTrace();  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //dbcourse\_seriesName 테이블에서 사용자가 입력한 시리즈 삭제  stmt.executeUpdate("delete from DBCOURSE\_Employee where id='"+emp\_ID+"'");  PreparedStatement pStmt=conn.prepareStatement("insert into DBCOURSE\_Employee values(?,?,?,?,?)");  pStmt.setString(1, emp\_ID);  pStmt.setString(2, newname);  pStmt.setString(3, newresname);  pStmt.setString(4, newposition);  pStmt.setInt(5, newsalary);  pStmt.executeUpdate();  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

* Should include queries that use view

|  |  |
| --- | --- |
| //delivery\_O view의 모든 tuple을 받아오기  ResultSet rs = stmt.executeQuery("select \* from DBCOURSE\_Delivery\_O"); | [cs](http://colorscripter.com/info#e) |
| //delivery\_X view의 모든 tuple을 받아오기  ResultSet rs = stmt.executeQuery("select \* from DBCOURSE\_Delivery\_X"); | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //score 기준을 선택했을 때  if(score\_radio.isSelected()){      //delivery인 경우      if(flag == 0) S = "select restaurant\_name, city, type, food\_name, price " +              "from (select restaurant\_name, city, type, food\_name, price, avg(score) as avg " +              "from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Evaluation " +              "group by food\_name) as A where avg";      //No delivery인 경우      else S = "select restaurant\_name, city, type, food\_name, price " +              "from (select restaurant\_name, city, type, food\_name, price, avg(score) as avg " +              "from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Evaluation " +              "group by food\_name) as A where avg";  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //region 기준을 선택했을 때  else if(region\_radio.isSelected()) {      /\*\* sql - 신유진\*/      //delivery인 경우      if(flag == 0) S = "select restaurant\_name, city, type, food\_name, price " +              "from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Address";      //No delivery인 경우      else S = "select restaurant\_name, city, type, food\_name, price " +              "from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Address"; | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //price 기준을 선택했을 때  else if(price\_radio.isSelected()) {      //price 조건을 제대로 선택한 경우 success label을 보여준다      select.setVisible(false);      try\_again.setVisible(false);      success.setVisible(true);      //delivery인 경우      if(flag == 0) S = "select \* from DBCOURSE\_Delivery\_O";      //No delivery인 경우      else S = "select \* from DBCOURSE\_Delivery\_X"; | [cs](http://colorscripter.com/info#e) |

|  |  |  |
| --- | --- | --- |
| while(rs.next()) { //선택된 모든 ingredient를 하나씩 빼는 while문      count++;      if(count == 1) {//처음 체크된 경우에만 where문 추가          /\*\* sql - 김해리, 신유진\*/          if(flag == 0) S += " where food\_name not in " +                  "(select DBCOURSE\_Delivery\_O.food\_name from DBCOURSE\_Delivery\_O, DBCOURSE\_Ingredient " +                  "where DBCOURSE\_Delivery\_O.food\_name = DBCOURSE\_Ingredient.food\_name " +                  "and (DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";          else S += S += " where food\_name not in " +                  "(select DBCOURSE\_Delivery\_X.food\_name from DBCOURSE\_Delivery\_X, DBCOURSE\_Ingredient " +                  "where DBCOURSE\_Delivery\_X.food\_name = DBCOURSE\_Ingredient.food\_name " +                  "and (DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";      }      else S += " or DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | | [cs](http://colorscripter.com/info#e) |
| //recommend 기준을 선택한 경우  else if(recommend\_radio.isSelected()) {      /\*\* sql - 김해리\*/      //delivery인 경우      if(flag == 0) S = "select restaurant\_name, city, type, food\_name, price " +              "from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Evaluation";      //No delivery인 경우      else S = "select restaurant\_name, city, type, food\_name, price " +              "from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Evaluation"; | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //accompany 기준을 선택한 경우  else if(accompany\_radio.isSelected()) {      /\*\* sql - 김해리, 신유진, 박소연\*/      //delivery인 경우 - only for not delivery의 조건이므로 빈 table과 try again 보여준다      if(flag == 0) {          S = "select A.restaurant\_name, A.city, A.type, A.food\_name, A.price " +                  "from (select \* from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Service)as A, " +                  "(select \* from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Service) as B " +                  "where A.restaurant\_name = B.restaurant\_name";          select.setVisible(false);          success.setVisible(false);          try\_again.setVisible(true);      }      //No delivery인 경우      else {          S = "select A.restaurant\_name, A.city, A.type, A.food\_name, A.price " +                  "from (select \* from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Service)as A, " +                  "(select \* from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Service) as B " +                  "where A.restaurant\_name = B.restaurant\_name";  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //parking 기준을 선택한 경우  else if(park\_radio.isSelected()) {      /\*\* sql - 박소연\*/      //delivery인 경우 - only for not delivery의 조건이므로 빈 table과 try again 보여준다      if(flag == 0) {          S = "select \* from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Address";          select.setVisible(false);          success.setVisible(false);          try\_again.setVisible(true);      }      //No delivery인 경우      else {          //선택한 조건에 따라 where문을 String S에 추가한다          S = "select \* from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Address"; | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //방금 들어간 주문의 재료를 가져온다  rs = stmt.executeQuery("select ingredient " +          "from DBCOURSE\_Order\_Ingredient " +          "where order\_number >= all(select order\_number from DBCOURSE\_Order\_Ingredient)"); | [cs](http://colorscripter.com/info#e) |

* Should include queries with nested queries

|  |  |
| --- | --- |
| //score 기준을 선택했을 때  if(score\_radio.isSelected()){      //delivery인 경우      if(flag == 0) S = "select restaurant\_name, city, type, food\_name, price " +              "from (select restaurant\_name, city, type, food\_name, price, avg(score) as avg " +              "from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Evaluation " +              "group by food\_name) as A where avg";      //No delivery인 경우      else S = "select restaurant\_name, city, type, food\_name, price " +              "from (select restaurant\_name, city, type, food\_name, price, avg(score) as avg " +              "from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Evaluation " +              "group by food\_name) as A where avg";  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |  |
| --- | --- | --- |
| while(rs.next()) { //선택된 모든 ingredient를 하나씩 빼는 while문      count++;      if(count == 1) {//처음 체크된 경우에만 where문 추가          /\*\* sql - 김해리, 신유진\*/          if(flag == 0) S += " where food\_name not in " +                  "(select DBCOURSE\_Delivery\_O.food\_name from DBCOURSE\_Delivery\_O, DBCOURSE\_Ingredient " +                  "where DBCOURSE\_Delivery\_O.food\_name = DBCOURSE\_Ingredient.food\_name " +                  "and (DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";          else S += S += " where food\_name not in " +                  "(select DBCOURSE\_Delivery\_X.food\_name from DBCOURSE\_Delivery\_X, DBCOURSE\_Ingredient " +                  "where DBCOURSE\_Delivery\_X.food\_name = DBCOURSE\_Ingredient.food\_name " +                  "and (DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";      }      else S += " or DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | | [cs](http://colorscripter.com/info#e) |
| //accompany 기준을 선택한 경우  else if(accompany\_radio.isSelected()) {      /\*\* sql - 김해리, 신유진, 박소연\*/      //delivery인 경우 - only for not delivery의 조건이므로 빈 table과 try again 보여준다      if(flag == 0) {          S = "select A.restaurant\_name, A.city, A.type, A.food\_name, A.price " +                  "from (select \* from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Service)as A, " +                  "(select \* from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Service) as B " +                  "where A.restaurant\_name = B.restaurant\_name";          select.setVisible(false);          success.setVisible(false);          try\_again.setVisible(true);      }      //No delivery인 경우      else {          S = "select A.restaurant\_name, A.city, A.type, A.food\_name, A.price " +                  "from (select \* from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Service)as A, " +                  "(select \* from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Service) as B " +                  "where A.restaurant\_name = B.restaurant\_name";  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //음식의 이름이 foodname일때 재료를 구해서 numIngredient가 0인 재료의 개수를 센다  String S = "select count(ingredient\_name) " +          "from (select DBCOURSE\_Ingredient.food\_name, DBCOURSE\_Inventory.ingredient\_name, DBCOURSE\_Inventory.numIngredient " +          "from DBCOURSE\_Inventory, DBCOURSE\_Ingredient, DBCOURSE\_Food " +          "where (DBCOURSE\_Ingredient.food\_name = DBCOURSE\_Food.name) " +          "and (DBCOURSE\_Inventory.ingredient\_name = DBCOURSE\_Ingredient.name) " +          "and(DBCOURSE\_Inventory.restaurant\_name = DBCOURSE\_Food.restaurant\_name)) as A " +          "where numIngredient = 0 and food\_name = '" + foodname + "'";  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //방금 들어간 주문의 재료를 가져온다  rs = stmt.executeQuery("select ingredient " +          "from DBCOURSE\_Order\_Ingredient " +          "where order\_number >= all(select order\_number from DBCOURSE\_Order\_Ingredient)"); | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //order number가 가장 작은 tuple을 고른다  ResultSet rs = stmt.executeQuery("select restaurant\_name, price from DBCOURSE\_Order " +          "where order\_number <= all(select order\_number from DBCOURSE\_Order)");  if(rs.next()) {      res = rs.getString(1);      price = rs.getInt(2);  }  //가져온 restaurant name과 price로 profit을 update한다  stmt.executeUpdate("update DBCOURSE\_Restaurant " +          "set profit = profit + " + price +          " where name = '" + res + "'");  //처리가 끝난 order를 order table에서 delete한다  stmt2.executeUpdate("delete from DBCOURSE\_Order " +          "where order\_number <= all(select order\_number from (select order\_number from DBCOURSE\_Order) as A)"); | [cs](http://colorscripter.com/info#e) |

* Should include queries with join queries

|  |  |
| --- | --- |
| //score 기준을 선택했을 때  if(score\_radio.isSelected()){      //delivery인 경우      if(flag == 0) S = "select restaurant\_name, city, type, food\_name, price " +              "from (select restaurant\_name, city, type, food\_name, price, avg(score) as avg " +              "from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Evaluation " +              "group by food\_name) as A where avg";      //No delivery인 경우      else S = "select restaurant\_name, city, type, food\_name, price " +              "from (select restaurant\_name, city, type, food\_name, price, avg(score) as avg " +              "from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Evaluation " +              "group by food\_name) as A where avg";  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //region 기준을 선택했을 때  else if(region\_radio.isSelected()) {      /\*\* sql - 신유진\*/      //delivery인 경우      if(flag == 0) S = "select restaurant\_name, city, type, food\_name, price " +              "from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Address";      //No delivery인 경우      else S = "select restaurant\_name, city, type, food\_name, price " +              "from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Address"; | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| while(rs.next()) { //선택된 모든 ingredient를 하나씩 빼는 while문      count++;      if(count == 1) {//처음 체크된 경우에만 where문 추가          /\*\* sql - 김해리, 신유진\*/          if(flag == 0) S += " where food\_name not in " +                  "(select DBCOURSE\_Delivery\_O.food\_name from DBCOURSE\_Delivery\_O, DBCOURSE\_Ingredient " +                  "where DBCOURSE\_Delivery\_O.food\_name = DBCOURSE\_Ingredient.food\_name " +                  "and (DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";          else S += S += " where food\_name not in " +                  "(select DBCOURSE\_Delivery\_X.food\_name from DBCOURSE\_Delivery\_X, DBCOURSE\_Ingredient " +                  "where DBCOURSE\_Delivery\_X.food\_name = DBCOURSE\_Ingredient.food\_name " +                  "and (DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";      }      else S += " or DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //recommend 기준을 선택한 경우  else if(recommend\_radio.isSelected()) {      /\*\* sql - 김해리\*/      //delivery인 경우      if(flag == 0) S = "select restaurant\_name, city, type, food\_name, price " +              "from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Evaluation";      //No delivery인 경우      else S = "select restaurant\_name, city, type, food\_name, price " +              "from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Evaluation"; | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //accompany 기준을 선택한 경우  else if(accompany\_radio.isSelected()) {      /\*\* sql - 김해리, 신유진, 박소연\*/      //delivery인 경우 - only for not delivery의 조건이므로 빈 table과 try again 보여준다      if(flag == 0) {          S = "select A.restaurant\_name, A.city, A.type, A.food\_name, A.price " +                  "from (select \* from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Service)as A, " +                  "(select \* from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Service) as B " +                  "where A.restaurant\_name = B.restaurant\_name";          select.setVisible(false);          success.setVisible(false);          try\_again.setVisible(true);      }      //No delivery인 경우      else {          S = "select A.restaurant\_name, A.city, A.type, A.food\_name, A.price " +                  "from (select \* from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Service)as A, " +                  "(select \* from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Service) as B " +                  "where A.restaurant\_name = B.restaurant\_name";  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //parking 기준을 선택한 경우  else if(park\_radio.isSelected()) {      /\*\* sql - 박소연\*/      //delivery인 경우 - only for not delivery의 조건이므로 빈 table과 try again 보여준다      if(flag == 0) {          S = "select \* from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Address";          select.setVisible(false);          success.setVisible(false);          try\_again.setVisible(true);      }      //No delivery인 경우      else {          //선택한 조건에 따라 where문을 String S에 추가한다          S = "select \* from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Address"; | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //음식의 이름이 foodname일때 재료를 구해서 numIngredient가 0인 재료의 개수를 센다  String S = "select count(ingredient\_name) " +          "from (select DBCOURSE\_Ingredient.food\_name, DBCOURSE\_Inventory.ingredient\_name, DBCOURSE\_Inventory.numIngredient " +          "from DBCOURSE\_Inventory, DBCOURSE\_Ingredient, DBCOURSE\_Food " +          "where (DBCOURSE\_Ingredient.food\_name = DBCOURSE\_Food.name) " +          "and (DBCOURSE\_Inventory.ingredient\_name = DBCOURSE\_Ingredient.name) " +          "and(DBCOURSE\_Inventory.restaurant\_name = DBCOURSE\_Food.restaurant\_name)) as A " +          "where numIngredient = 0 and food\_name = '" + foodname + "'";  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

* Should include queries that are parameterized and dynamically created (in other words, takes input value from user and creates query using the user given value)

|  |  |  |
| --- | --- | --- |
| else if(e.getSource()==enterinfobutton){      //새로운 정보를 textfield와 choice로부터 받아온다.      newname=name.getText();      newposition=c\_position.getSelectedItem();      newresname=c\_res\_name.getSelectedItem();      String newsalary\_s=salary.getText(); | | [cs](http://colorscripter.com/info#e) |
| //salary를 up하는 경우  else if(e.getSource()==givebutton\_up){      try{          String str=plus\_salary.getText();          Double value=Double.valueOf(str); | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //salary를 down하는 경우  else if(e.getSource()==givebutton\_down){      //textfield로부터 String을 받아온다.      try{          String str=minus\_salary.getText();          Double value=Double.valueOf(str); | [cs](http://colorscripter.com/info#e) |

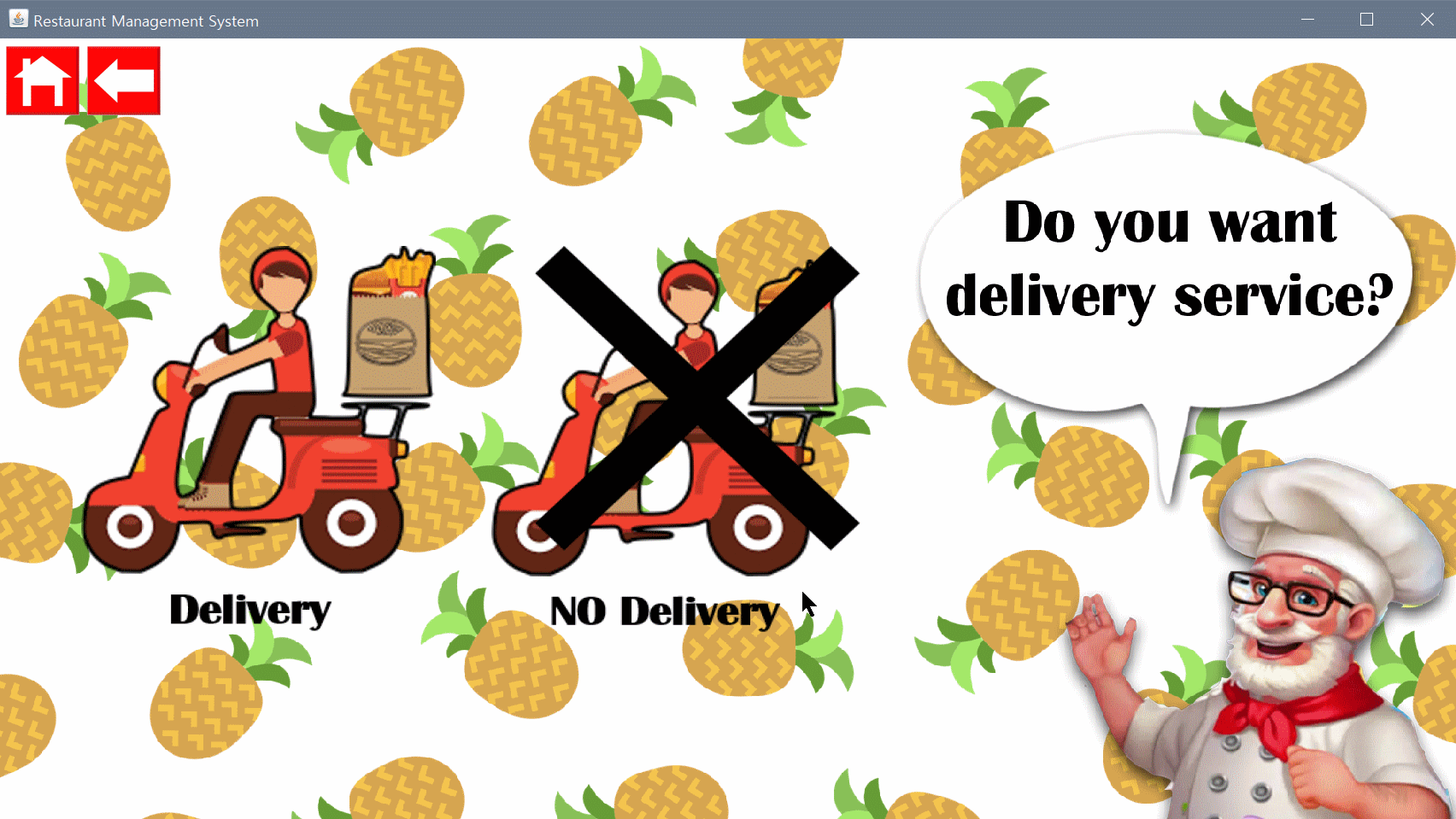
|  |  |
| --- | --- |
| //salary를 기준으로 직원을 해고하는 경우  else if(e.getSource()==firebutton){      //textfield로부터 string을 받아온다      try{          String str=delete\_salary.getText();          Double value=Double.valueOf(str); | [cs](http://colorscripter.com/info#e) |

* Should have user interface which is graphical or text-based. And should have a menu that is easy to use and has helpful information for usage. Easy use of program will be given consideration in evaluation

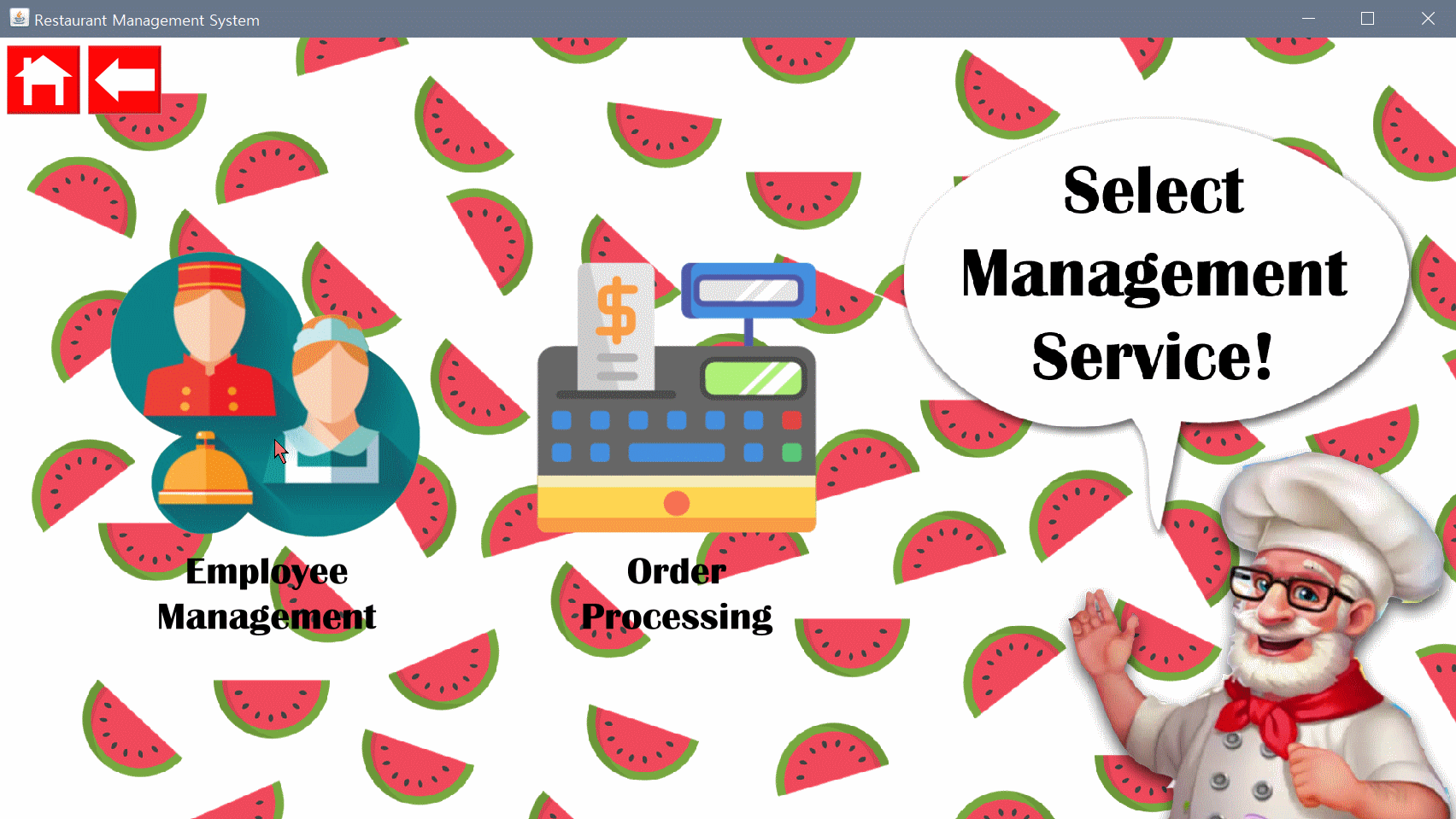
<First Main Page – Select Manager mode or User mode>



<User page 1 – Select Delivery service or No Delivery service>

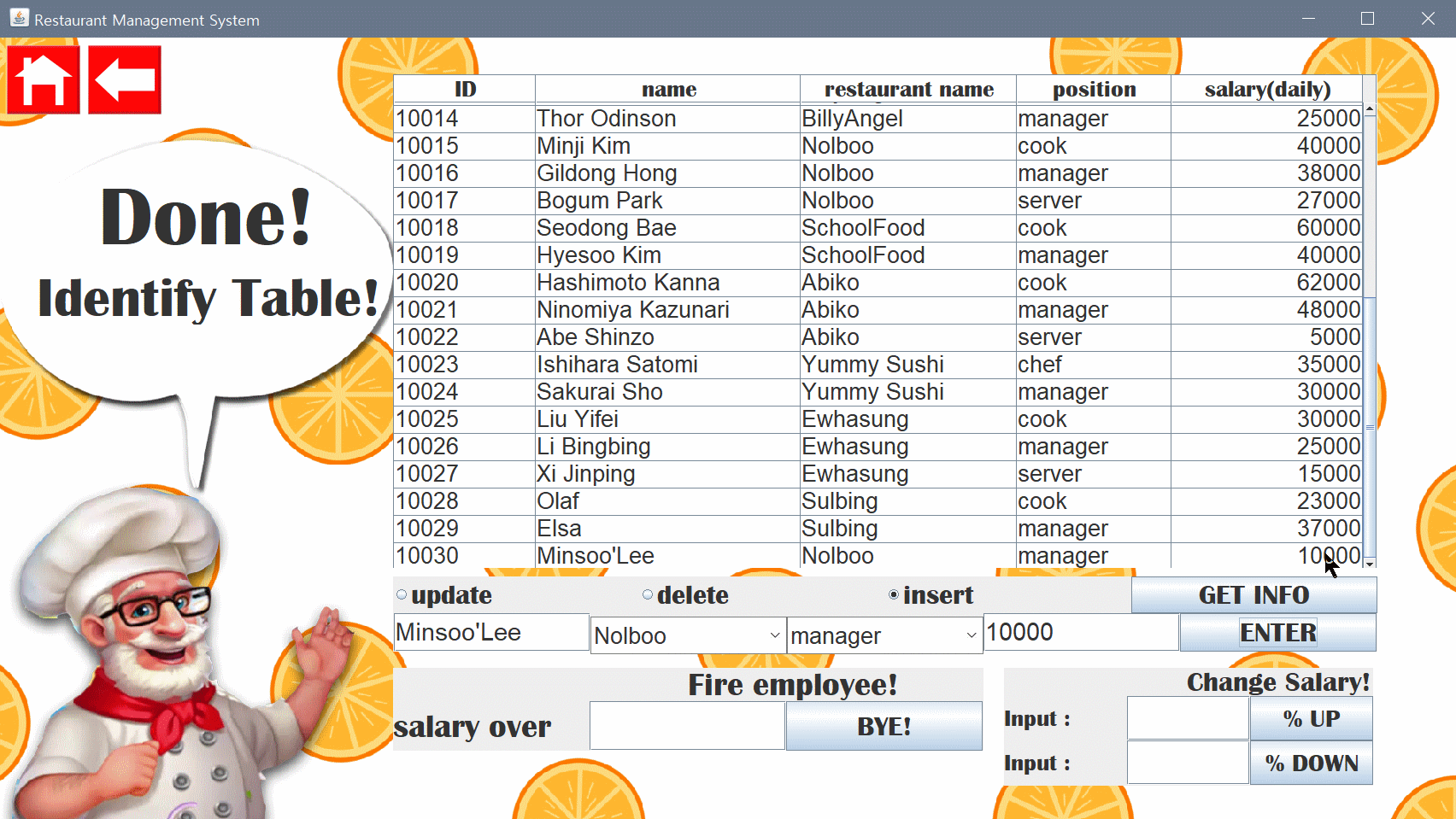
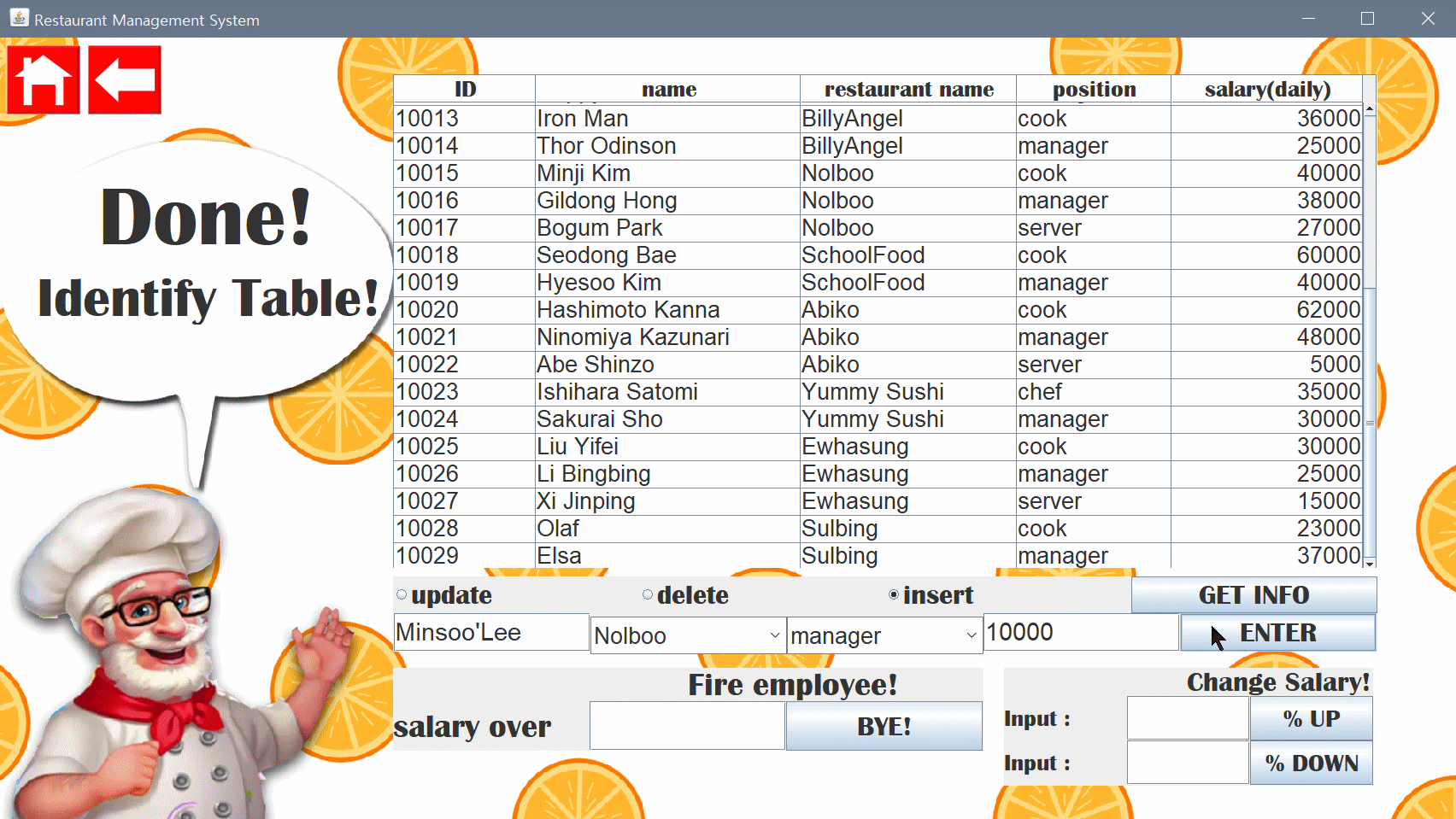


<Manager Page 1 – Select Employee Management mode or Order Processing mode>



* Should have interface and queries to insert into database

<Insert Employee>



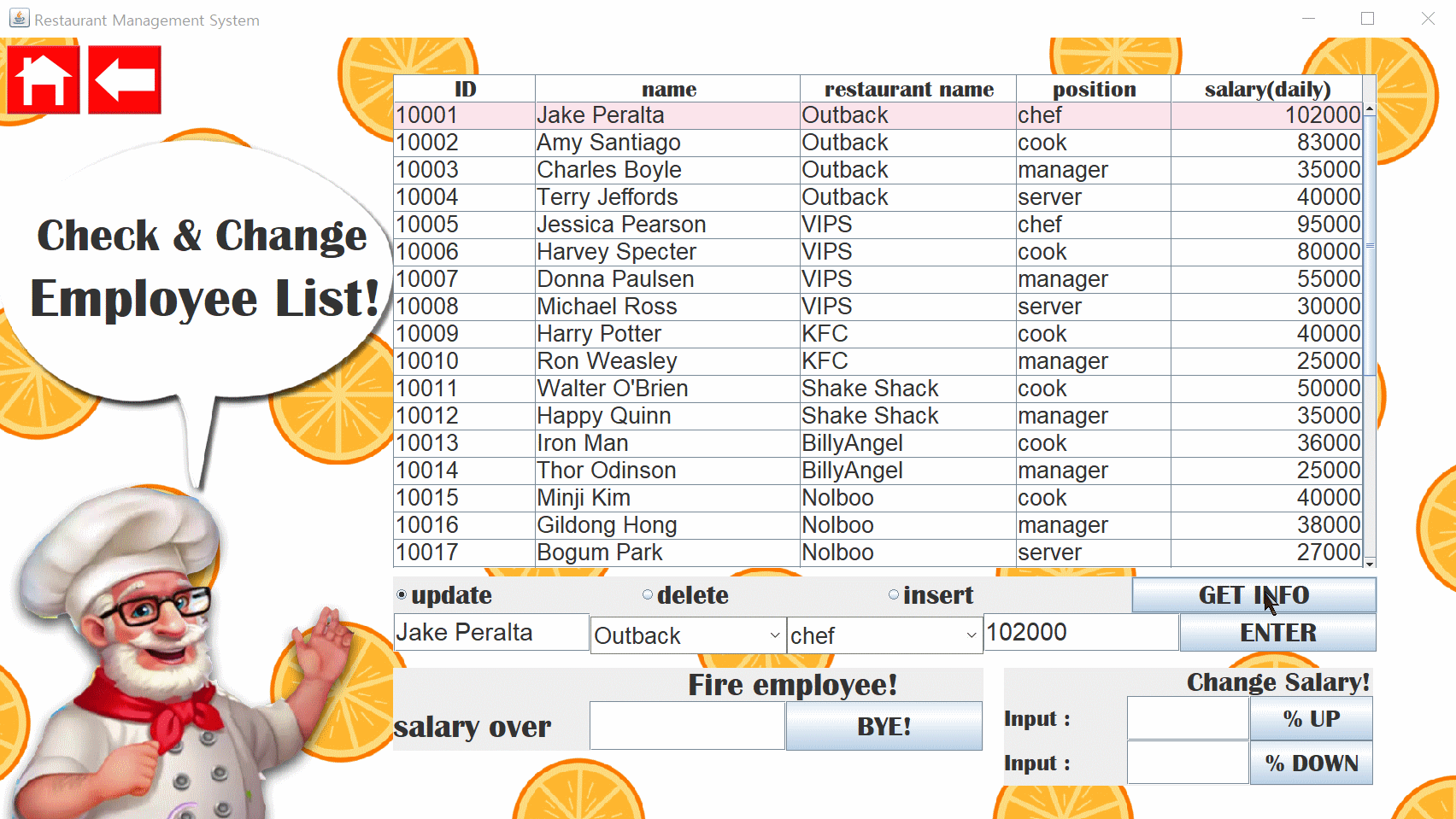
|  |  |
| --- | --- |
| try {      /\*\* 박소연\*/      //DBCOURSE\_Order에 insert를 할 때 preparedstatement 이용      PreparedStatement pStmt=conn.prepareStatement("insert into DBCOURSE\_Order values(?,?,?,?)");      rs = stmt.executeQuery("select max(order\_number) from DBCOURSE\_Order");        /\*\* 배소연\*/      //주문번호는 max(order\_number)+1로 자동 셋팅      if(rs.next()) count = rs.getInt(1);      count++;        /\*\* 박소연\*/      //delivery인 경우      if(flag==0){          pStmt.setInt(1, count);          pStmt.setString(2, res\_name);          pStmt.setString(3, foodname);          //음식가격에 배달료를 더하여 음식가격을 설정          pStmt.setInt(4, (Integer.valueOf(foodprice)+Integer.valueOf(deliverycost)));          pStmt.executeUpdate();      }      //No delivery인 경우      else{          pStmt.setInt(1, count);          pStmt.setString(2, res\_name);          pStmt.setString(3, foodname);          pStmt.setInt(4, (Integer.valueOf(foodprice)));          pStmt.executeUpdate();      }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

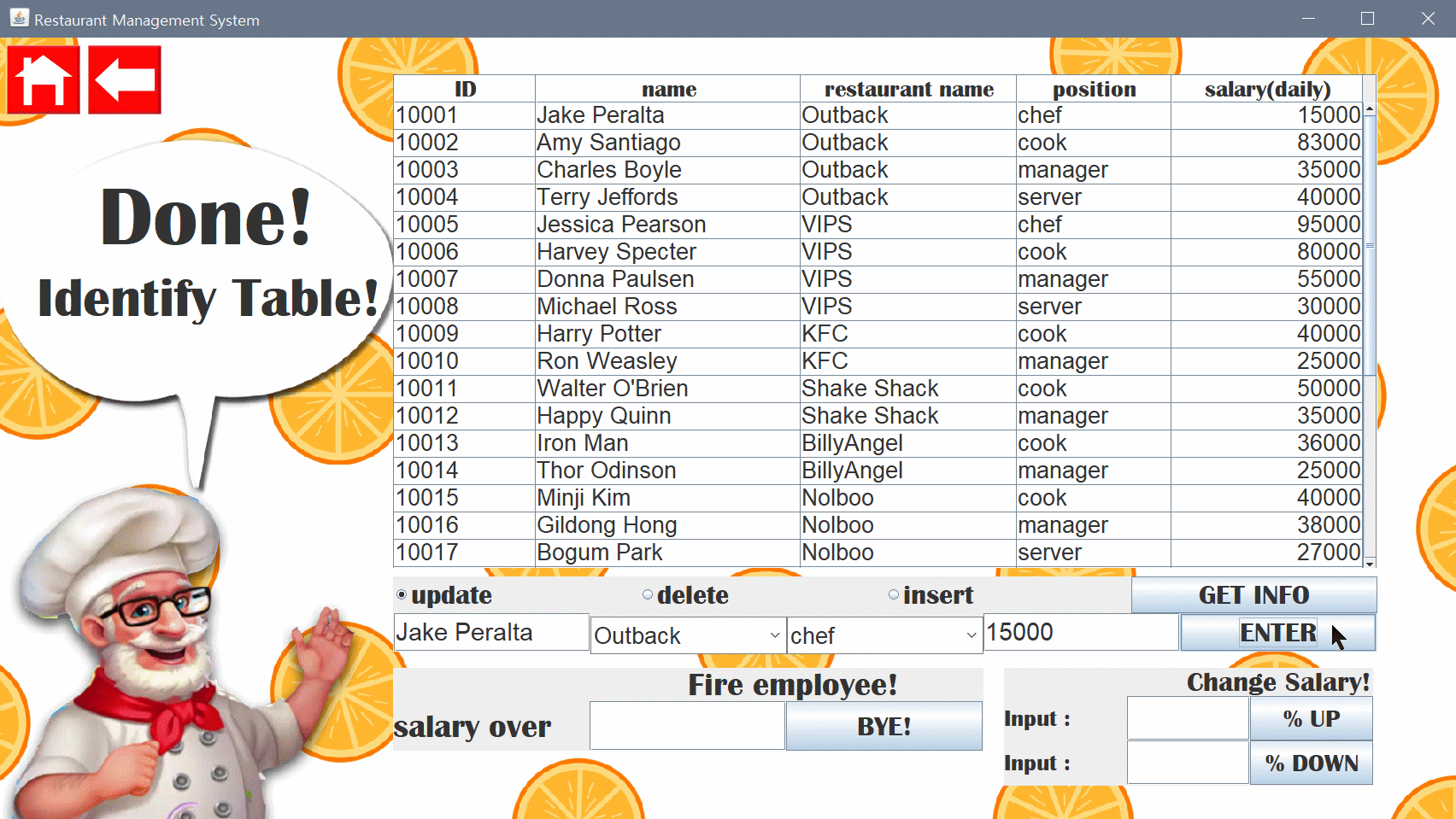
|  |  |
| --- | --- |
| //새로운 정보를 table에 insert한다  //DBCOURSE\_Employee에 prepared statement를 만든다  PreparedStatement pStmt=conn.prepareStatement("insert into DBCOURSE\_Employee values(?,?,?,?,?)");  pStmt.setString(1, String.valueOf(newid+1));  pStmt.setString(2, newname);  pStmt.setString(3, newresname);  pStmt.setString(4, newposition);  pStmt.setInt(5, newsalary);  pStmt.executeUpdate();  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //dbcourse\_seriesName 테이블에서 사용자가 입력한 시리즈 삭제  stmt.executeUpdate("delete from DBCOURSE\_Employee where id='"+emp\_ID+"'");  PreparedStatement pStmt=conn.prepareStatement("insert into DBCOURSE\_Employee values(?,?,?,?,?)");  pStmt.setString(1, emp\_ID);  pStmt.setString(2, newname);  pStmt.setString(3, newresname);  pStmt.setString(4, newposition);  pStmt.setInt(5, newsalary);  pStmt.executeUpdate();  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

* Should have interface and queries to update from database

<Update Employee>



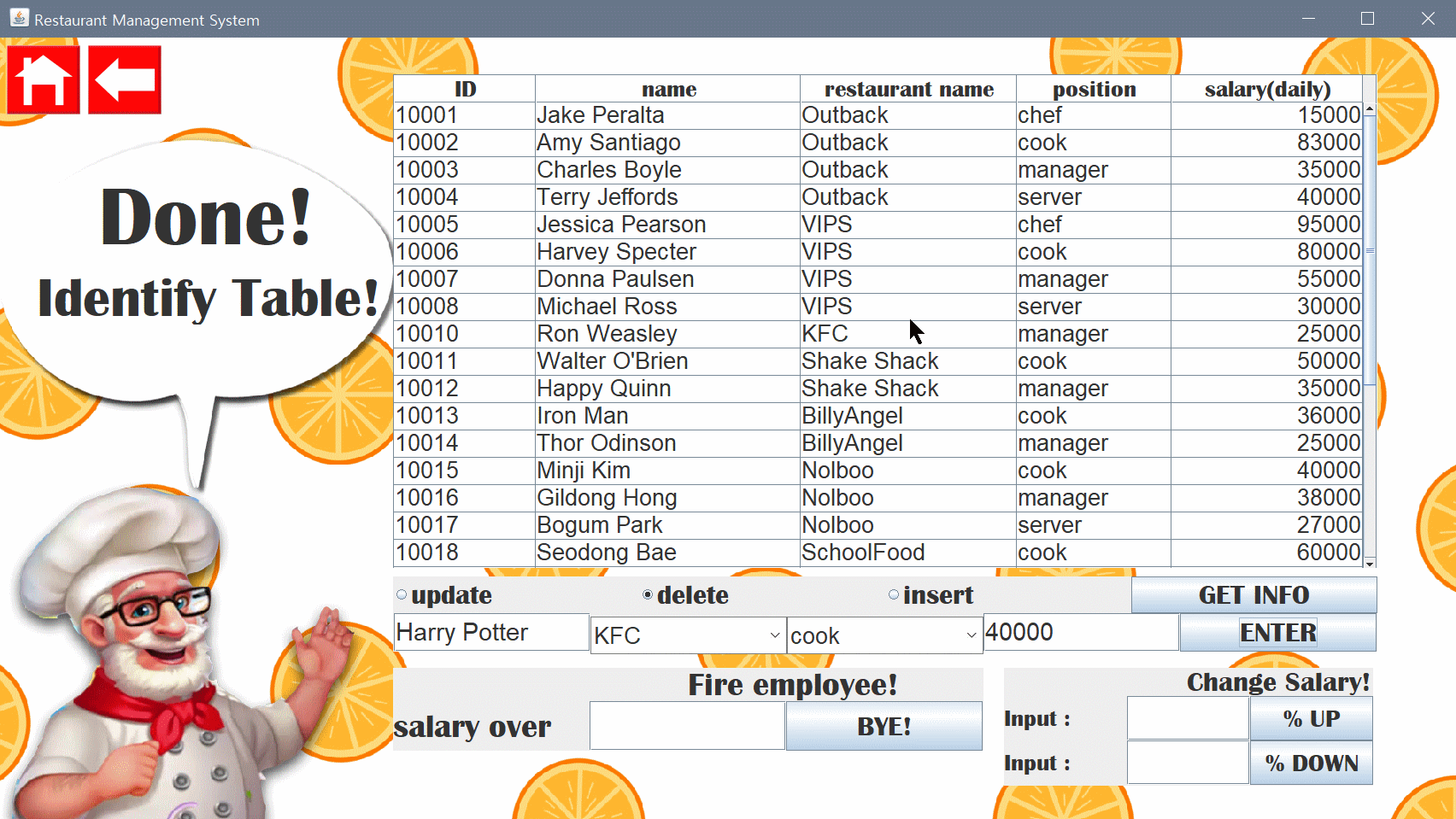
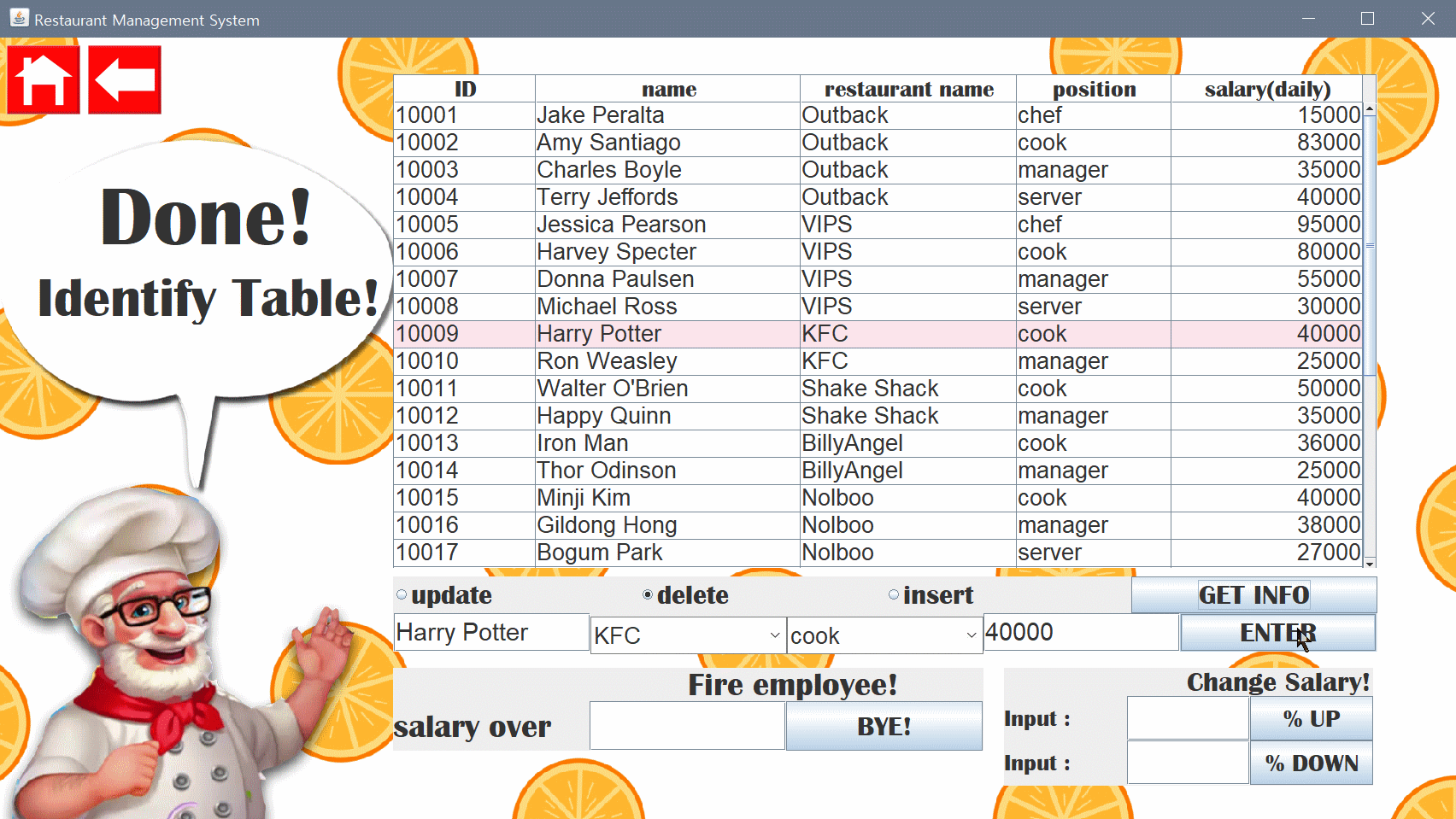


|  |  |
| --- | --- |
| while(rs.next()) { //방금 들어간 주문의 레스토랑에서 재료를 하나씩 빼서 요리한다.      ingr = rs.getString(1);      stmt2.executeUpdate("update DBCOURSE\_Inventory " +              "set numIngredient = numIngredient - 1 " +              "where DBCOURSE\_Inventory.restaurant\_name = '" + res\_name +              "' and DBCOURSE\_Inventory.ingredient\_name = '" + ingr + "' ");  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |  |  |
| --- | --- | --- | --- |
| //salary 결과를 update한다.  String S="update DBCOURSE\_Employee set salary=salary\*"+String.valueOf(1+Double.valueOf(str)/(double)100);  try {      stmt.executeUpdate(S);  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | | | [cs](http://colorscripter.com/info#e) |
| else{ //0과 100 사이의 수라면 salary update      String S="update DBCOURSE\_Employee set salary=salary\*"+String.valueOf(1-Double.valueOf(str)/(double)100);      try {          stmt.executeUpdate(S);  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | | | | [cs](http://colorscripter.com/info#e) |
| //가져온 restaurant name과 price로 profit을 update한다  stmt.executeUpdate("update DBCOURSE\_Restaurant " +          "set profit = profit + " + price +          " where name = '" + res + "'"); | | [cs](http://colorscripter.com/info#e) |
| while(rs.next()) { //가져온 restaurant name과 sum(salary)로 profit을 update한다      res = rs.getString(1); sum = rs.getInt(2);      stmt2.executeUpdate("update DBCOURSE\_Restaurant " +              "set profit = profit - " + sum +              " where name = '" + res + "'");  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

* Should have interface and queries to delete from database

<Delete Employee>



|  |  |
| --- | --- |
| //employee id로 찾은 employee 정보를 DBCOURSE\_Employee에서 제거한다  try{      stmt.executeUpdate("delete from DBCOURSE\_Employee where id='"+emp\_ID+"'");  } catch(SQLException se) {      se.printStackTrace();  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

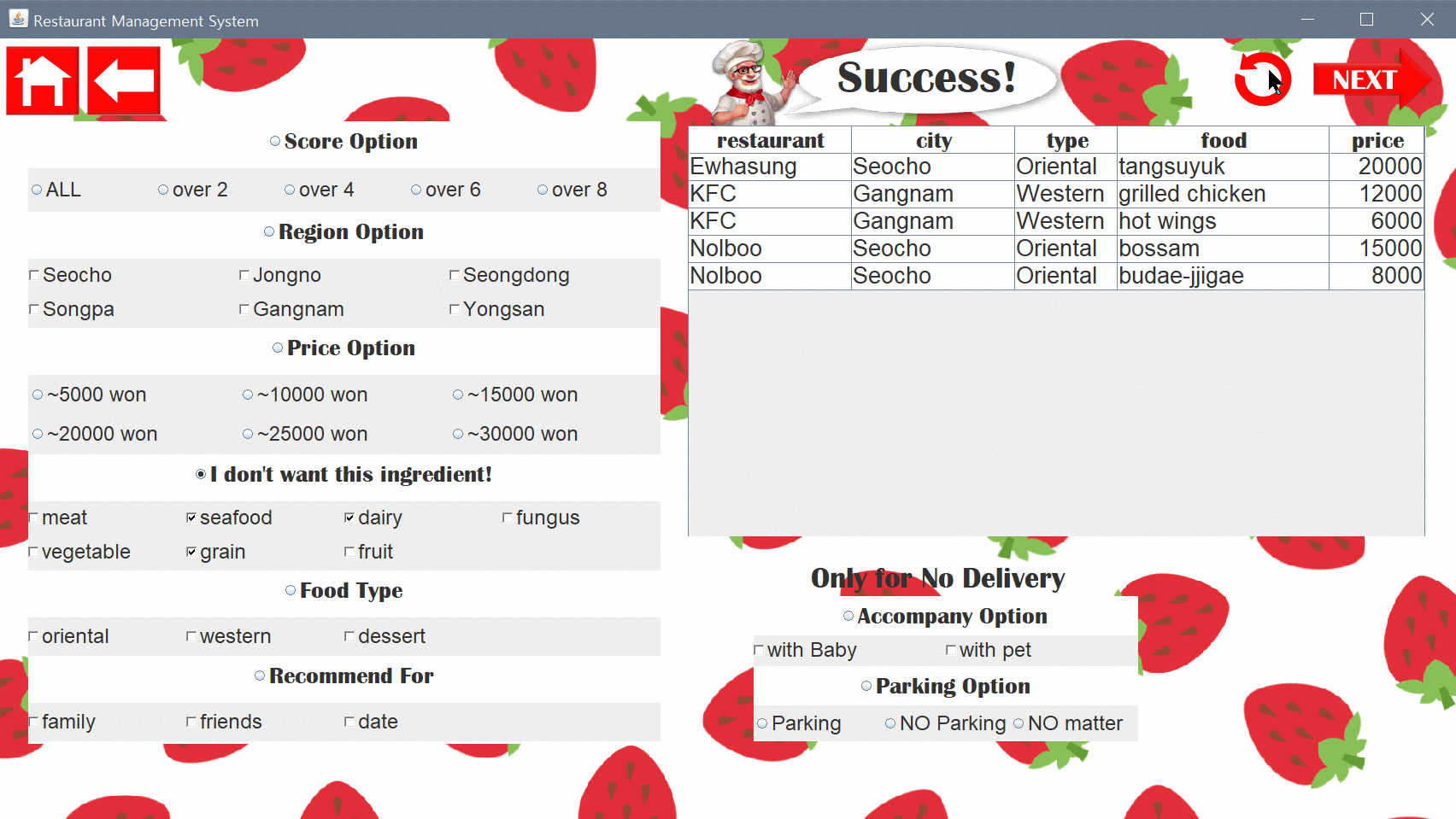
|  |  |
| --- | --- |
| //dbcourse\_seriesName 테이블에서 사용자가 입력한 시리즈 삭제  stmt.executeUpdate("delete from DBCOURSE\_Employee where id='"+emp\_ID+"'"); | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| String S="delete from DBCOURSE\_Employee where salary>"+String.valueOf(value);    //textfield에 입력된 수보다 큰 salary를 가진 employee는 delete한다.  try {      stmt.executeUpdate(S);  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //처리가 끝난 order를 order table에서 delete한다  stmt2.executeUpdate("delete from DBCOURSE\_Order " +          "where order\_number <= all(select order\_number from (select order\_number from DBCOURSE\_Order) as A)"); | [cs](http://colorscripter.com/info#e) |

* Should have interface and queries to select from database

<Select condition of limitation>



|  |  |
| --- | --- |
| try {      //DBCOURSE\_Employee로부터 정보를 가져온다.      ResultSet rs = stmt.executeQuery("select \* from DBCOURSE\_Employee");      while(rs.next()) {          Object data[] = {rs.getString(1), rs.getString(2), rs.getString(3), rs.getString(4), rs.getInt(5)};          model.addRow(data);      }  } catch (SQLException e1) {      e1.printStackTrace();  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| try {      //order table의 모든 tuple 받아오기      ResultSet rs = stmt.executeQuery("select \* from DBCOURSE\_Order");      //한 tuple씩 받아온다      while(rs.next()) {          Object data[] = {rs.getInt(1), rs.getString(2), rs.getString(3), rs.getInt(4)};          model.addRow(data);      }      //restaurant table에서 모든 restaurant\_name과 profit 받아오기      rs = stmt.executeQuery("select name as restaurant\_name, profit from DBCOURSE\_Restaurant");      //한 줄씩 읽는다      while(rs.next()) {          Object data[] = {rs.getString(1), rs.getInt(2)};          model2.addRow(data);      }  } catch (SQLException e1) {      e1.printStackTrace();  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| try {      //delivery\_O view의 모든 tuple을 받아오기      ResultSet rs = stmt.executeQuery("select \* from DBCOURSE\_Delivery\_O");      //한 tuple씩 받아온다      while(rs.next()) {          Object data[] = {rs.getString(1), rs.getString(2), rs.getString(3), rs.getString(4), rs.getInt(5)};          model.addRow(data);      }  } catch (SQLException e1) {      e1.printStackTrace();  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| try {      //delivery\_X view의 모든 tuple을 받아오기      ResultSet rs = stmt.executeQuery("select \* from DBCOURSE\_Delivery\_X");      //한 tuple씩 받아온다      while(rs.next()) {          Object data[] = {rs.getString(1), rs.getString(2), rs.getString(3), rs.getString(4), rs.getInt(5)};          model.addRow(data);      }  } catch (SQLException e1) {      e1.printStackTrace();  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //score 기준을 선택했을 때  if(score\_radio.isSelected()){      //delivery인 경우      if(flag == 0) S = "select restaurant\_name, city, type, food\_name, price " +              "from (select restaurant\_name, city, type, food\_name, price, avg(score) as avg " +              "from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Evaluation " +              "group by food\_name) as A where avg";      //No delivery인 경우      else S = "select restaurant\_name, city, type, food\_name, price " +              "from (select restaurant\_name, city, type, food\_name, price, avg(score) as avg " +              "from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Evaluation " +              "group by food\_name) as A where avg";  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //region 기준을 선택했을 때  else if(region\_radio.isSelected()) {      /\*\* sql - 신유진\*/      //delivery인 경우      if(flag == 0) S = "select restaurant\_name, city, type, food\_name, price " +              "from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Address";      //No delivery인 경우      else S = "select restaurant\_name, city, type, food\_name, price " +              "from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Address"; | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //price 기준을 선택했을 때  else if(price\_radio.isSelected()) {      //price 조건을 제대로 선택한 경우 success label을 보여준다      select.setVisible(false);      try\_again.setVisible(false);      success.setVisible(true);      //delivery인 경우      if(flag == 0) S = "select \* from DBCOURSE\_Delivery\_O";      //No delivery인 경우      else S = "select \* from DBCOURSE\_Delivery\_X"; | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //remove ingredient 기준을 선택했을 때  else if(remove\_radio.isSelected()) {      /\*\* 배소연\*/      //delivery인 경우      if(flag == 0) S = "select \* from DBCOURSE\_Delivery\_O";      //No delivery인 경우      else S = "select \* from DBCOURSE\_Delivery\_X";      //다중 선택 체크박스 배열      Checkbox[] check = {meat, seafood, dairy, fungus, vegetable, grain, fruit};      int count = 0; //선택된 체크박스의 개수와 선택된 ingredient 개수 세는 count 변수      String S2 = "select name from DBCOURSE\_Ingredient ";      for(int i=0; i<check.length; i++) {          if(check[i].getState() == true) {              count++;              if(count == 1) S2 += "where type = '" + check[i].getLabel() + "'";              else S2 += " or type = '" + check[i].getLabel() + "'";          }      }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| while(rs.next()) { //선택된 모든 ingredient를 하나씩 빼는 while문      count++;      if(count == 1) {//처음 체크된 경우에만 where문 추가          /\*\* sql - 김해리, 신유진\*/          if(flag == 0) S += " where food\_name not in " +                  "(select DBCOURSE\_Delivery\_O.food\_name from DBCOURSE\_Delivery\_O, DBCOURSE\_Ingredient " +                  "where DBCOURSE\_Delivery\_O.food\_name = DBCOURSE\_Ingredient.food\_name " +                  "and (DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";          else S += S += " where food\_name not in " +                  "(select DBCOURSE\_Delivery\_X.food\_name from DBCOURSE\_Delivery\_X, DBCOURSE\_Ingredient " +                  "where DBCOURSE\_Delivery\_X.food\_name = DBCOURSE\_Ingredient.food\_name " +                  "and (DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";      }      else S += " or DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //recommend 기준을 선택한 경우  else if(recommend\_radio.isSelected()) {      /\*\* sql - 김해리\*/      //delivery인 경우      if(flag == 0) S = "select restaurant\_name, city, type, food\_name, price " +              "from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Evaluation";      //No delivery인 경우      else S = "select restaurant\_name, city, type, food\_name, price " +              "from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Evaluation"; | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //accompany 기준을 선택한 경우  else if(accompany\_radio.isSelected()) {      /\*\* sql - 김해리, 신유진, 박소연\*/      //delivery인 경우 - only for not delivery의 조건이므로 빈 table과 try again 보여준다      if(flag == 0) {          S = "select A.restaurant\_name, A.city, A.type, A.food\_name, A.price " +                  "from (select \* from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Service)as A, " +                  "(select \* from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Service) as B " +                  "where A.restaurant\_name = B.restaurant\_name";          select.setVisible(false);          success.setVisible(false);          try\_again.setVisible(true);      }      //No delivery인 경우      else {          S = "select A.restaurant\_name, A.city, A.type, A.food\_name, A.price " +                  "from (select \* from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Service)as A, " +                  "(select \* from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Service) as B " +                  "where A.restaurant\_name = B.restaurant\_name";  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //parking 기준을 선택한 경우  else if(park\_radio.isSelected()) {      /\*\* sql - 박소연\*/      //delivery인 경우 - only for not delivery의 조건이므로 빈 table과 try again 보여준다      if(flag == 0) {          S = "select \* from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Address";          select.setVisible(false);          success.setVisible(false);          try\_again.setVisible(true);      }      //No delivery인 경우      else {          //선택한 조건에 따라 where문을 String S에 추가한다          S = "select \* from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Address"; | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //delivery인 경우  if(flag==0){      try{          //레스토랑의 delivery cost를 받아온다          String S = "select delivery\_cost from DBCOURSE\_Delivery where restaurant\_name='"+res\_name+"'";          ResultSet rs = stmt.executeQuery(S);          while(rs.next()){              deliverycost=String.valueOf(rs.getInt(1));              delivery\_cost.setText(deliverycost+" won");          }          delivery\_cost\_label.setText("Delivery Cost: ");      } catch(SQLException e1){          e1.printStackTrace();      }  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //음식의 이름이 foodname일때 재료를 구해서 numIngredient가 0인 재료의 개수를 센다  String S = "select count(ingredient\_name) " +          "from (select DBCOURSE\_Ingredient.food\_name, DBCOURSE\_Inventory.ingredient\_name, DBCOURSE\_Inventory.numIngredient " +          "from DBCOURSE\_Inventory, DBCOURSE\_Ingredient, DBCOURSE\_Food " +          "where (DBCOURSE\_Ingredient.food\_name = DBCOURSE\_Food.name) " +          "and (DBCOURSE\_Inventory.ingredient\_name = DBCOURSE\_Ingredient.name) " +          "and(DBCOURSE\_Inventory.restaurant\_name = DBCOURSE\_Food.restaurant\_name)) as A " +          "where numIngredient = 0 and food\_name = '" + foodname + "'";  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| rs = stmt.executeQuery("select max(order\_number) from DBCOURSE\_Order"); | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //방금 들어간 주문의 재료를 가져온다  rs = stmt.executeQuery("select ingredient " +          "from DBCOURSE\_Order\_Ingredient " +          "where order\_number >= all(select order\_number from DBCOURSE\_Order\_Ingredient)"); | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| try {      //레스토랑의 전화번호를 가져온다.      ResultSet rs = stmt.executeQuery("select phone\_number from DBCOURSE\_restaurant where name="+"'"+res\_name+"'");      while(rs.next()) {          res\_phone=rs.getString(1);      }  } catch (SQLException e1) {      e1.printStackTrace();  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| try {      //이전 아이디에 +1 함으로써 새로운 아이디를 자동으로 부여한다      int newid=0;      ResultSet rs1 = stmt.executeQuery("select max(id) from DBCOURSE\_Employee");      if(rs1.next()) newid = Integer.valueOf(rs1.getString(1)); | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| ResultSet rs;  try {      rs = stmt.executeQuery("SELECT \* FROM DBCOURSE\_Employee");      while(rs.next()) {          Object data[] = {rs.getString(1), rs.getString(2), rs.getString(3), rs.getString(4), rs.getInt(5)};          model.addRow(data);      }  } catch (SQLException e1) {      // TODO Auto-generated catch block      e1.printStackTrace();  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //order number가 가장 작은 tuple을 고른다  ResultSet rs = stmt.executeQuery("select restaurant\_name, price from DBCOURSE\_Order " +          "where order\_number <= all(select order\_number from DBCOURSE\_Order)");  if(rs.next()) {      res = rs.getString(1);      price = rs.getInt(2);  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //처리가 끝난 order를 order table에서 delete한다  stmt2.executeUpdate("delete from DBCOURSE\_Order " +          "where order\_number <= all(select order\_number from (select order\_number from DBCOURSE\_Order) as A)");    //남은 order table의 tuple을 가져온다  rs = stmt.executeQuery("select \* from DBCOURSE\_Order");  while(rs.next()) {      Object data[] = {rs.getInt(1), rs.getString(2), rs.getString(3), rs.getInt(4)};      model.addRow(data);  }  //update된 restaurant 별 현재의 profit을 가져온다  rs = stmt.executeQuery("select name as restaurant\_name, profit from DBCOURSE\_Restaurant");  while(rs.next()) {      Object data[] = {rs.getString(1), rs.getInt(2)};      model2.addRow(data);  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //영업 종료를 하고 직원들에게 daily salary를 지급하고 최종 profit을 계산하는 버튼  else if(e.getSource() == closebutton) {      try {          int count = 0;          //order table이 다 처리 되어 비어 있는 상태인지 확인          ResultSet rs = stmt.executeQuery("select count(order\_number) from DBCOURSE\_Order");          if(rs.next()) count = rs.getInt(1);  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //레스토랑 별 sum(salary)를 구한다  rs = stmt.executeQuery("select restaurant\_name, sum(salary) " +          "from DBCOURSE\_Employee " +          "group by restaurant\_name");  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

|  |  |
| --- | --- |
| //update된 table을 보여준다  rs = stmt.executeQuery("select name as restaurant\_name, profit from DBCOURSE\_Restaurant");  while(rs.next()) {      Object data[] = {rs.getString(1), rs.getInt(2)};      model2.addRow(data);  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

1. **Responsibility**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Member name | SQL | Java code | Report | Presentation | Demo Video | Etc. |
| 김해리 | **createdb.sql**  **create table and insert, modify**  -DBCOURSE\_Restaurant  -DBCOURSE\_Ingredient  -DBCOURSE\_Food  -DBCOURSE\_Employee  -DBCOURSE\_Evaluation  -DBCOURSE\_Address  -DBCOURSE\_Delivery  -DBCOURSE\_Service  -DBCOURSE\_Order  -DBCOURSE\_Inventory  **create view**  -DBCOURSE\_Delivery\_O  -DBCOURSE\_Delivery\_X  **create ER Diagram** | **actionPerformed**  -Create SQL to meet the condition in score, remove ingredient, type, accompany, recommend of updatebutton  -Subtract daily salary of each restaurant’s employees from total profit | -Database Schema Diagram  -Main class name and How to run, Connection configuration  -Capture executing program  -17 requirements  -Responsibility | -Presentation | -Record video  -Create demo video  -Modify scenario | -Create and edit background images  -Create button images |
| 박소연 | **created.sql**  **modify**  -DBCOURSE\_Restaurant  -DBCOURSE\_Food  -DBCOURSE\_Ingredient  -DBCOURSE\_Employee  -DBCOURSE\_Evaluation  -DBCOURSE\_Address  -DBCOURSE\_Service  -DBCOURSE\_Inventory  **create view**  -DBCOURSE\_Delivery\_O  -DBCOURSE\_Delivery\_X  -DBCOURSE\_Order\_Ingredient  **modify ER Diagram** | -annotate  **GUI**  -Frame window constructor  -Card layout  -create card in all page  -u1 page : JButton for delivery, no delivery  -u2 page : JRadioButton, Checkbox, JLabel for check list  -u3 page : JLabel for bill  -m2 page : JLabel to inform, JRadioButton, ButtonGroup, JButton in box for button and textfield, JTextField, Choice  **actionPerformed**  -Employee management part of manager mode  -updatebutton : write region jdbc code, price sql and jdbc code, type jdbc code, recommend jdbc code, accompany sql and jdbc code, parking sql and jdbc code  -nextbutton\_u2 : select phone\_number for selected restaurant  -getinfobutton : get tuple and show in JTextfield and choice, show JLabel to meet the conditions  -enterinfobutton : create sql and jdbc code to insert and delete employee tuple  -givebutton\_up, givebutton\_down : create sql and jdbc code to update salary  -firebutton : create delete sql and jdbc code to delete employee with salary condition  -Control NumberFormatExceoption | -Class and method explanation  -Capture executing program-Responsibility |  | -Write scenario | -Design contents and organize system  -Create button images  -Create and edit background images  -Create jar file |
| 배소연 | **created.sql**  **modify**  -DBCOURSE\_Food  -DBCOURSE\_Ingredient  -DBCOURSE\_Order  -DBCOURSE\_Employee  -DBCOURSE\_Address  **create view**  -DBCOURSE\_Delivery\_O  -DBCOURSE\_Delivery\_X  -DBCOURSE\_Order\_Ingredient  **confirm ER Diagram** | -Connect with mySQL  -annotate  **GUI**  -ImageIcon  -changeImageSize method  -draw background image in all page  -create JButton(home, back) in all page  -home page : create JButton for manager mode, user mode  -u2 page : JLabel to inform  -u3 page : JLabel to inform  -m1 page : JButton for employee management, order processing  -m2 page : show JTable, JLabel to inform, set font to JTextField  -m3 page : show JTable, JButton for take order and close, JLabel to inform  **actionPerformed**  -homebutton, backbutton, manager mode button, user mode button  -Order processing part of manager mode  -deliverbutton, nodeliverbutton  -updatebutton : show label when improper check, write score jdbc code, remove ingredient sql and jdbc code, accompany sql and jdbc code, parking sql  -Transaction control in JDBC – when order, handle checking whether numIngredient is not zero, when update tuple in employee management page, when take order and update restaurants’ profit  -nextbutton\_u2 : create sql and jdbc code to calculate numIngredient  -getinfobutton : show JLabel to meet the conditions  -enterinfobutton : show JTable to meet the conditions  -takeorderbutton : create sql, jdbc code to delete order table and update profit  -closebutton : create sql, jdbc code to update profit-make  -showTable method using JTable, DefaultTableModel, JScrollPane  -override getColumnClass to sort integer type in JTable | -Main class name and How to run, Connection configuration  -17 requirements  -Responsibility  -SQL Scripts  -Java codes |  | -Modify scenario  -Edit background music  -Create demo video | -Design contents and organize system  -Create background images  -Create button images  -Create jar file |
| 신유진 | **createdb.sql**  **create table and insert**  -DBCOURSE\_Restaurant  -DBCOURSE\_Ingredient  -DBCOURSE\_Food  -DBCOURSE\_Employee  -DBCOURSE\_Evaluation  -DBCOURSE\_Address  -DBCOURSE\_Delivery  -DBCOURSE\_Service  -DBCOURSE\_Order  -DBCOURSE\_Inventory  **create view**  -DBCOURSE\_Delivery\_O  -DBCOURSE\_Delivery\_X  **create index**  **dropdb.sql**  **confirm ER Diagram** | -Connect with mySQL  **actionPerformed**  -Create SQL to meet the condition of score, region, remove ingredient, accompany of updatebutton | -Responsibility | -Write script  -Presentation | -Scenario  -Subtitles of video  -Create demo video  -insert background music  -Write and modify scenario  -Edit background music | -Create button images  -Design contents and organize system  -Editing images |

1. **SQL Scripts**

-createdb.sql

|  |  |
| --- | --- |
| create database team1;  use team1;  SET FOREIGN\_KEY\_CHECKS=0;  SET sql\_mode=(SELECT REPLACE(@@sql\_mode,'ONLY\_FULL\_GROUP\_BY',''));    /\*  -Responsibility-  DBCOURSE\_Restaurant : 김해리, 박소연, 신유진  DBCOURSE\_Ingredient : 김해리, 신유진  DBCOURSE\_Food : 김해리, 박소연, 배소연, 신유진  DBCOURSE\_Employee : 김해리, 박소연, 배소연, 신유진  DBCOURSE\_Evaluation : 김해리, 박소연, 신유진  DBCOURSE\_Address : 김해리, 박소연, 배소연, 신유진  DBCOURSE\_Delivery : 김해리, 신유진  DBCOURSE\_Service : 김해리, 박소연, 신유진  DBCOURSE\_Order : 김해리, 배소연, 신유진  DBCOURSE\_Inventory : 김해리, 박소연, 신유진  DBCOURSE\_Delivery\_O : 김해리, 배소연, 신유진  DBCOURSE\_Delivery\_X : 김해리, 배소연, 신유진  DBCOURSE\_Order\_Ingredient : 배소연  create index : 신유진  dropdb.sql : 신유진  \*/    create table DBCOURSE\_Restaurant (      name varchar(20) primary key,      type varchar(20) not null,      phone\_number varchar(15) not null,      profit integer(10) not null);    insert into DBCOURSE\_Restaurant values ('Nolboo', 'Oriental', '02-557-3848', 0);  insert into DBCOURSE\_Restaurant values ('SchoolFood', 'Oriental', '02-510-5005', 0);  insert into DBCOURSE\_Restaurant values ('Abiko', 'Oriental', '02-514-3212', 0);  insert into DBCOURSE\_Restaurant values ('Yummy Sushi', 'Oriental', '02-545-0023', 0);  insert into DBCOURSE\_Restaurant values ('Outback', 'Western', '02-558-4491', 0);  insert into DBCOURSE\_Restaurant values ('VIPS', 'Western', '02-3442-2997', 0);  insert into DBCOURSE\_Restaurant values ('Ewhasung', 'Oriental', '02-393-8511', 0);  insert into DBCOURSE\_Restaurant values ('BillyAngel', 'Dessert', '02-512-6257', 0);  insert into DBCOURSE\_Restaurant values ('Sulbing', 'Dessert', '02-518-1415', 0);  insert into DBCOURSE\_Restaurant values ('KFC', 'Western', '02-547-1664', 0);  insert into DBCOURSE\_Restaurant values ('Shake Shack', 'Western', '02-510-5217', 0);    create table DBCOURSE\_Food (      name varchar(20) primary key,      type varchar(20) not null,      price integer(10) not null,      restaurant\_name varchar(20) not null,      foreign key(restaurant\_name) references DBCOURSE\_Restaurant(name));    insert into DBCOURSE\_Food values ('bossam', 'Oriental', 15000, 'Nolboo');  insert into DBCOURSE\_Food values ('budae-jjigae', 'Oriental', 8000, 'Nolboo');  insert into DBCOURSE\_Food values ('wheat-tteokbokki','Oriental', 4000, 'SchoolFood');  insert into DBCOURSE\_Food values ('rice-tteokbokki','Oriental',4000, 'SchoolFood');  insert into DBCOURSE\_Food values ('tuna gimbap', 'Oriental',3000, 'SchoolFood');  insert into DBCOURSE\_Food values ('mushroom curry rice', 'Oriental',8000, 'Abiko');  insert into DBCOURSE\_Food values ('pork curry rice', 'Oriental',8000, 'Abiko');  insert into DBCOURSE\_Food values ('salmon roll','Oriental',9000, 'Yummy Sushi');  insert into DBCOURSE\_Food values ('tuna roll','Oriental',9000, 'Yummy Sushi');  insert into DBCOURSE\_Food values ('gyoza','Oriental',6000, 'Yummy Sushi');  insert into DBCOURSE\_Food values ('T-bone steak','Western',28000, 'Outback');  insert into DBCOURSE\_Food values ('ribs steak','Western',25000, 'Outback');  insert into DBCOURSE\_Food values ('Pork-sirloin steak','Western',24000, 'Outback');  insert into DBCOURSE\_Food values ('Beef-sirloin steak','Western',27000, 'VIPS');  insert into DBCOURSE\_Food values ('tomato pasta','Western',13000, 'VIPS');  insert into DBCOURSE\_Food values ('carbonara','Western',12000, 'VIPS');  insert into DBCOURSE\_Food values ('seafood pasta','Western',15000, 'VIPS');  insert into DBCOURSE\_Food values ('shrimp fried rice','Oriental',11000, 'Ewhasung');  insert into DBCOURSE\_Food values ('jjamppong','Oriental',8000, 'Ewhasung');  insert into DBCOURSE\_Food values ('tangsuyuk','Oriental',20000, 'Ewhasung');  insert into DBCOURSE\_Food values ('jajangmyeon','Oriental',8000, 'Ewhasung');  insert into DBCOURSE\_Food values ('tiramisu','Dessert',6000, 'BillyAngel');  insert into DBCOURSE\_Food values ('strawberry tart','Dessert',5000, 'BillyAngel');  insert into DBCOURSE\_Food values ('cheese cake','Dessert',5000, 'BillyAngel');  insert into DBCOURSE\_Food values ('mango snow ice','Dessert',7000, 'Sulbing');  insert into DBCOURSE\_Food values ('injeolmi snow ice','Dessert',8000, 'Sulbing');  insert into DBCOURSE\_Food values ('chicken burger','Western',8000, 'KFC');  insert into DBCOURSE\_Food values ('hot wings','Western',6000, 'KFC');  insert into DBCOURSE\_Food values ('grilled chicken','Western',12000, 'KFC');  insert into DBCOURSE\_Food values ('shack burger','Western',8000, 'Shake Shack');  insert into DBCOURSE\_Food values ('mushroom burger','Western',7000, 'Shake Shack');  insert into DBCOURSE\_Food values ('hot dog','Western',6000, 'Shake Shack');  insert into DBCOURSE\_Food values ('vanila shakes','Western',5000, 'Shake Shack');    create table DBCOURSE\_Ingredient (      name varchar(20) not null,      food\_name varchar(20) not null,      type varchar(20) not null,     primary key(name, food\_name),     foreign key(food\_name) references DBCOURSE\_Food(name)  );    insert into DBCOURSE\_Ingredient values ('pork', 'bossam', 'meat');  insert into DBCOURSE\_Ingredient values ('pork', 'pork curry rice', 'meat');  insert into DBCOURSE\_Ingredient values ('pork', 'ribs steak', 'meat');  insert into DBCOURSE\_Ingredient values ('pork', 'Pork-siroin steak', 'meat');  insert into DBCOURSE\_Ingredient values ('pork', 'tangsuyuk', 'meat');  insert into DBCOURSE\_Ingredient values ('beef', 'T-bone steak','meat');  insert into DBCOURSE\_Ingredient values ('beef', 'Beef-siroin steak','meat');  insert into DBCOURSE\_Ingredient values ('sausage', 'budae-jjigae', 'meat');  insert into DBCOURSE\_Ingredient values ('sausage', 'hot dog', 'meat');  insert into DBCOURSE\_Ingredient values ('bacon', 'shack burger', 'meat');  insert into DBCOURSE\_Ingredient values ('tuna', 'tuna gimbap', 'seafood');  insert into DBCOURSE\_Ingredient values ('tuna', 'tuna roll', 'seafood');  insert into DBCOURSE\_Ingredient values ('milk', 'carbonara', 'dairy');  insert into DBCOURSE\_Ingredient values ('milk', 'mango snow ice', 'dairy');  insert into DBCOURSE\_Ingredient values ('milk', 'injeolmi snow ice', 'dairy');  insert into DBCOURSE\_Ingredient values ('milk', 'vanila shakes', 'dairy');  insert into DBCOURSE\_Ingredient values ('shrimp', 'seafood pasta', 'seafood');  insert into DBCOURSE\_Ingredient values ('shrimp', 'shrimp fried rice', 'seafood');  insert into DBCOURSE\_Ingredient values ('shrimp', 'jjamppong', 'seafood');  insert into DBCOURSE\_Ingredient values ('salmon', 'salmon roll', 'seafood');  insert into DBCOURSE\_Ingredient values ('mushroom', 'mushroom curry rice', 'fungus');  insert into DBCOURSE\_Ingredient values ('mushroom', 'carbonara', 'fungus');  insert into DBCOURSE\_Ingredient values ('mushroom', 'mushroom burger', 'fungus');  insert into DBCOURSE\_Ingredient values ('tomato', 'tomato pasta', 'vegetable');  insert into DBCOURSE\_Ingredient values ('carrot', 'tuna gimbap', 'vegetable');  insert into DBCOURSE\_Ingredient values ('kimchi', 'bossam', 'vegetable');  insert into DBCOURSE\_Ingredient values ('kimchi', 'budae-jjigae', 'vegetable');  insert into DBCOURSE\_Ingredient values ('chicken', 'chicken burger', 'meat');  insert into DBCOURSE\_Ingredient values ('chicken', 'hot wings', 'meat');  insert into DBCOURSE\_Ingredient values ('chicken', 'grilled chicken', 'meat');  insert into DBCOURSE\_Ingredient values ('rice', 'rice-tteokbokki', 'grain');  insert into DBCOURSE\_Ingredient values ('rice', 'tuna gimbap', 'grain');  insert into DBCOURSE\_Ingredient values ('rice', 'mushroom curry rice', 'grain');  insert into DBCOURSE\_Ingredient values ('rice', 'pork curry rice', 'grain');  insert into DBCOURSE\_Ingredient values ('rice', 'salmon roll', 'grain');  insert into DBCOURSE\_Ingredient values ('rice', 'tuna roll', 'grain');  insert into DBCOURSE\_Ingredient values ('rice', 'shrimp fried rice', 'grain');  insert into DBCOURSE\_Ingredient values ('rice', 'injeolmi snow ice', 'grain');  insert into DBCOURSE\_Ingredient values ('wheat', 'wheat-tteokbokki', 'grain');  insert into DBCOURSE\_Ingredient values ('wheat', 'gyoza', 'grain');  insert into DBCOURSE\_Ingredient values ('wheat', 'tomato pasta', 'grain');  insert into DBCOURSE\_Ingredient values ('wheat', 'carbonara', 'grain');  insert into DBCOURSE\_Ingredient values ('wheat', 'seafood pasta', 'grain');  insert into DBCOURSE\_Ingredient values ('wheat', 'jjamppong', 'grain');  insert into DBCOURSE\_Ingredient values ('wheat', 'jajangmyeon', 'grain');  insert into DBCOURSE\_Ingredient values ('wheat', 'strawberry tart', 'grain');  insert into DBCOURSE\_Ingredient values ('wheat', 'cheese cake', 'grain');  insert into DBCOURSE\_Ingredient values ('wheat', 'chicken burger', 'grain');  insert into DBCOURSE\_Ingredient values ('wheat', 'shack burger', 'grain');  insert into DBCOURSE\_Ingredient values ('wheat', 'mushroom burger', 'grain');  insert into DBCOURSE\_Ingredient values ('wheat', 'hot dog', 'grain');  insert into DBCOURSE\_Ingredient values ('squid', 'seafood pasta', 'seafood');  insert into DBCOURSE\_Ingredient values ('squid', 'jjamppong', 'seafood');  insert into DBCOURSE\_Ingredient values ('strawberry', 'strawberry tart', 'fruit');  insert into DBCOURSE\_Ingredient values ('cheese', 'cheese cake', 'dairy');  insert into DBCOURSE\_Ingredient values ('mango', 'mango snow ice', 'fruit');  insert into DBCOURSE\_Ingredient values ('lettuce', 'chicken burger', 'vegetable');  insert into DBCOURSE\_Ingredient values ('lettuce', 'shack burger', 'vegetable');  insert into DBCOURSE\_Ingredient values ('lettuce', 'mushroom burger', 'vegetable');    create table DBCOURSE\_Employee (      id char(5) primary key,      name varchar(20) not null,      restaurant\_name varchar(20) not null,      position varchar(20) not null,      salary integer(10) not null,      foreign key(restaurant\_name) references DBCOURSE\_Restaurant(name));    insert into DBCOURSE\_Employee values ('10001','Jake Peralta', 'Outback', 'chef', 102000);  insert into DBCOURSE\_Employee values ('10002','Amy Santiago', 'Outback', 'cook', 83000);  insert into DBCOURSE\_Employee values ('10003','Charles Boyle', 'Outback', 'manager', 35000);  insert into DBCOURSE\_Employee values ('10004','Terry Jeffords', 'Outback', 'server', 40000);  insert into DBCOURSE\_Employee values ('10005','Jessica Pearson', 'VIPS', 'chef', 95000);  insert into DBCOURSE\_Employee values ('10006','Harvey Specter', 'VIPS', 'cook', 80000);  insert into DBCOURSE\_Employee values ('10007','Donna Paulsen', 'VIPS', 'manager', 55000);  insert into DBCOURSE\_Employee values ('10008','Michael Ross', 'VIPS', 'server', 30000);  insert into DBCOURSE\_Employee values ('10009','Harry Potter', 'KFC', 'cook', 40000);  insert into DBCOURSE\_Employee values ('10010','Ron Weasley', 'KFC', 'manager', 25000);  insert into DBCOURSE\_Employee values ('10011','Walter O\'Brien', 'Shake Shack', 'cook', 50000);  insert into DBCOURSE\_Employee values ('10012','Happy Quinn', 'Shake Shack', 'manager', 35000);  insert into DBCOURSE\_Employee values ('10013','Iron Man', 'BillyAngel', 'cook', 36000);  insert into DBCOURSE\_Employee values ('10014','Thor Odinson', 'BillyAngel', 'manager', 25000);  insert into DBCOURSE\_Employee values ('10015','Minji Kim', 'Nolboo', 'cook', 40000);  insert into DBCOURSE\_Employee values ('10016','Gildong Hong', 'Nolboo', 'manager', 38000);  insert into DBCOURSE\_Employee values ('10017','Bogum Park', 'Nolboo', 'server', 27000);  insert into DBCOURSE\_Employee values ('10018','Seodong Bae', 'SchoolFood', 'cook', 60000);  insert into DBCOURSE\_Employee values ('10019','Hyesoo Kim', 'SchoolFood', 'manager', 40000);  insert into DBCOURSE\_Employee values ('10020','Hashimoto Kanna', 'Abiko', 'cook', 62000);  insert into DBCOURSE\_Employee values ('10021','Ninomiya Kazunari', 'Abiko', 'manager', 48000);  insert into DBCOURSE\_Employee values ('10022','Abe Shinzo', 'Abiko', 'server', 5000);  insert into DBCOURSE\_Employee values ('10023','Ishihara Satomi', 'Yummy Sushi', 'chef', 35000);  insert into DBCOURSE\_Employee values ('10024','Sakurai Sho', 'Yummy Sushi', 'manager', 30000);  insert into DBCOURSE\_Employee values ('10025','Liu Yifei', 'Ewhasung', 'cook', 30000);  insert into DBCOURSE\_Employee values ('10026','Li Bingbing', 'Ewhasung', 'manager', 25000);  insert into DBCOURSE\_Employee values ('10027','Xi Jinping', 'Ewhasung', 'server', 15000);  insert into DBCOURSE\_Employee values ('10028','Olaf', 'Sulbing', 'cook', 23000);  insert into DBCOURSE\_Employee values ('10029','Elsa', 'Sulbing', 'manager', 37000);    create table DBCOURSE\_Evaluation (      restaurant\_name varchar(20) not null,      score integer(10) not null,      recommendation varchar(20) not null,      foreign key(restaurant\_name) references DBCOURSE\_Restaurant(name),      primary key(restaurant\_name, score, recommendation));    insert into DBCOURSE\_Evaluation values ('Nolboo', 8, 'family');  insert into DBCOURSE\_Evaluation values ('Nolboo', 5, 'family');  insert into DBCOURSE\_Evaluation values ('Nolboo', 10, 'friends');  insert into DBCOURSE\_Evaluation values ('SchoolFood', 2, 'family');  insert into DBCOURSE\_Evaluation values ('SchoolFood', 1, 'date');  insert into DBCOURSE\_Evaluation values ('Abiko', 2, 'friends');  insert into DBCOURSE\_Evaluation values ('Abiko', 2, 'family');  insert into DBCOURSE\_Evaluation values ('Abiko', 1, 'friends');  insert into DBCOURSE\_Evaluation values ('Yummy Sushi', 10, 'date');  insert into DBCOURSE\_Evaluation values ('Yummy Sushi', 8, 'family');  insert into DBCOURSE\_Evaluation values ('Yummy Sushi', 6, 'date');  insert into DBCOURSE\_Evaluation values ('Yummy Sushi', 9, 'friends');  insert into DBCOURSE\_Evaluation values ('Outback', 10, 'date');  insert into DBCOURSE\_Evaluation values ('Outback', 5, 'friends');  insert into DBCOURSE\_Evaluation values ('VIPS', 3, 'date');  insert into DBCOURSE\_Evaluation values ('VIPS', 5, 'friends');  insert into DBCOURSE\_Evaluation values ('Ewhasung', 6, 'friends');  insert into DBCOURSE\_Evaluation values ('Ewhasung', 9, 'family');  insert into DBCOURSE\_Evaluation values ('Ewhasung', 5, 'friends');  insert into DBCOURSE\_Evaluation values ('BillyAngel', 7, 'date');  insert into DBCOURSE\_Evaluation values ('BillyAngel', 9, 'date');  insert into DBCOURSE\_Evaluation values ('BillyAngel', 3, 'friends');  insert into DBCOURSE\_Evaluation values ('Sulbing', 6, 'friends');  insert into DBCOURSE\_Evaluation values ('Sulbing', 4, 'date');  insert into DBCOURSE\_Evaluation values ('Sulbing', 5, 'family');  insert into DBCOURSE\_Evaluation values ('KFC', 4, 'friends');  insert into DBCOURSE\_Evaluation values ('KFC', 3, 'family');  insert into DBCOURSE\_Evaluation values ('Shake Shack', 8, 'friends');  insert into DBCOURSE\_Evaluation values ('Shake Shack', 7, 'date');  insert into DBCOURSE\_Evaluation values ('Shake Shack', 10, 'friends');  insert into DBCOURSE\_Evaluation values ('Shake Shack', 8, 'family');    create table DBCOURSE\_Address (      restaurant\_name varchar(20) primary key,      city varchar(10) not null,      parking char(1),      foreign key(restaurant\_name) references DBCOURSE\_Restaurant(name));    insert into DBCOURSE\_Address values ('Nolboo','Seocho', null);  insert into DBCOURSE\_Address values ('SchoolFood','Jongno', null);  insert into DBCOURSE\_Address values ('Abiko','Jongno','o');  insert into DBCOURSE\_Address values ('Yummy Sushi','Seongdong', null);  insert into DBCOURSE\_Address values ('Outback','Yongsan','o');  insert into DBCOURSE\_Address values ('VIPS','Songpa','x');  insert into DBCOURSE\_Address values ('Ewhasung','Seocho', null);  insert into DBCOURSE\_Address values ('BillyAngel','Gangnam','o');  insert into DBCOURSE\_Address values ('Sulbing','Yongsan', null);  insert into DBCOURSE\_Address values ('KFC','Gangnam', null);  insert into DBCOURSE\_Address values ('Shake Shack','Gangnam','x');    create table DBCOURSE\_Delivery (      restaurant\_name varchar(20) primary key,      deliver char(1) not null,      delivery\_cost integer(10),      foreign key(restaurant\_name) references DBCOURSE\_Restaurant(name));    insert into DBCOURSE\_Delivery values ('Nolboo', 'o', 2000);  insert into DBCOURSE\_Delivery values ('SchoolFood', 'o', 0);  insert into DBCOURSE\_Delivery values ('Abiko', 'x', null);  insert into DBCOURSE\_Delivery values ('Yummy Sushi', 'o', 1000);  insert into DBCOURSE\_Delivery values ('Outback', 'x', null);  insert into DBCOURSE\_Delivery values ('VIPS', 'x', null);  insert into DBCOURSE\_Delivery values ('Ewhasung', 'o', 0);  insert into DBCOURSE\_Delivery values ('BillyAngel', 'x', null);  insert into DBCOURSE\_Delivery values ('Sulbing', 'o', 4000);  insert into DBCOURSE\_Delivery values ('KFC', 'o', 0);  insert into DBCOURSE\_Delivery values ('Shake Shack', 'x', null);    create table DBCOURSE\_Service (      restaurant\_name varchar(20) not null,      permission varchar(20) not null,      foreign key(restaurant\_name) references DBCOURSE\_Restaurant(name),      primary key(restaurant\_name, permission));    insert into DBCOURSE\_Service values ('Abiko','baby');  insert into DBCOURSE\_Service values ('Outback','pet');  insert into DBCOURSE\_Service values ('Outback','baby');  insert into DBCOURSE\_Service values ('VIPS','pet');  insert into DBCOURSE\_Service values ('BillyAngel','pet');  insert into DBCOURSE\_Service values ('BillyAngel','baby');  insert into DBCOURSE\_Service values ('Shake Shack','none');    create table DBCOURSE\_Order (      order\_number integer(10) not null,      restaurant\_name varchar(20) not null,      food\_name varchar(20) not null,      price integer(10) not null,      foreign key (restaurant\_name) references DBCOURSE\_Restaurant(name),      foreign key (food\_name) references DBCOURSE\_Food(name),      primary key(order\_number));    create table DBCOURSE\_Inventory(      restaurant\_name varchar(20) not null,      ingredient\_name varchar(20) not null,      numIngredient integer(10) not null,      foreign key(restaurant\_name) references DBCOURSE\_Restaurant(name),      foreign key(ingredient\_name) references DBCOURSE\_Ingredient(name),      primary key(restaurant\_name, ingredient\_name));    insert into DBCOURSE\_Inventory values ('Nolboo', 'kimchi', 10);  insert into DBCOURSE\_Inventory values ('Nolboo', 'sausage', 10);  insert into DBCOURSE\_Inventory values ('Nolboo', 'pork', 8);  insert into DBCOURSE\_Inventory values ('SchoolFood', 'wheat', 13);  insert into DBCOURSE\_Inventory values ('SchoolFood', 'rice', 9);  insert into DBCOURSE\_Inventory values ('SchoolFood', 'carrot', 1);  insert into DBCOURSE\_Inventory values ('SchoolFood', 'tuna', 2);  insert into DBCOURSE\_Inventory values ('Abiko', 'mushroom', 3);  insert into DBCOURSE\_Inventory values ('Abiko', 'rice', 5);  insert into DBCOURSE\_Inventory values ('Abiko', 'pork', 4);  insert into DBCOURSE\_Inventory values ('Yummy Sushi', 'salmon', 1);  insert into DBCOURSE\_Inventory values ('Yummy Sushi', 'tuna', 3);  insert into DBCOURSE\_Inventory values ('Yummy Sushi', 'rice', 10);  insert into DBCOURSE\_Inventory values ('Yummy Sushi', 'wheat', 10);  insert into DBCOURSE\_Inventory values ('Outback', 'beef', 3);  insert into DBCOURSE\_Inventory values ('Outback', 'pork', 5);  insert into DBCOURSE\_Inventory values ('VIPS', 'beef', 10);  insert into DBCOURSE\_Inventory values ('VIPS', 'wheat', 15);  insert into DBCOURSE\_Inventory values ('VIPS', 'tomato', 6);  insert into DBCOURSE\_Inventory values ('VIPS', 'milk', 6);  insert into DBCOURSE\_Inventory values ('VIPS', 'mushroom', 10);  insert into DBCOURSE\_Inventory values ('VIPS', 'shrimp', 7);  insert into DBCOURSE\_Inventory values ('VIPS', 'squid', 9);  insert into DBCOURSE\_Inventory values ('Ewhasung', 'wheat', 10);  insert into DBCOURSE\_Inventory values ('Ewhasung', 'rice', 7);  insert into DBCOURSE\_Inventory values ('Ewhasung', 'shrimp', 9);  insert into DBCOURSE\_Inventory values ('Ewhasung', 'squid', 0);  insert into DBCOURSE\_Inventory values ('Ewhasung', 'pork', 4);  insert into DBCOURSE\_Inventory values ('BillyAngel','wheat',10);  insert into DBCOURSE\_Inventory values ('BillyAngel','strawberry',8);  insert into DBCOURSE\_Inventory values ('BillyAngel','cheese',7);  insert into DBCOURSE\_Inventory values ('Sulbing','milk',10);  insert into DBCOURSE\_Inventory values ('Sulbing','mango',3);  insert into DBCOURSE\_Inventory values ('Sulbing','rice',7);  insert into DBCOURSE\_Inventory values ('KFC','chicken',10);  insert into DBCOURSE\_Inventory values ('KFC','lettuce',4);  insert into DBCOURSE\_Inventory values ('KFC','wheat',8);  insert into DBCOURSE\_Inventory values ('Shake Shack','bacon',10);  insert into DBCOURSE\_Inventory values ('Shake Shack','lettuce',10);  insert into DBCOURSE\_Inventory values ('Shake Shack','wheat',10);  insert into DBCOURSE\_Inventory values ('Shake Shack','mushroom',1);  insert into DBCOURSE\_Inventory values ('Shake Shack','sausage',10);  insert into DBCOURSE\_Inventory values ('Shake Shack','milk',10);    create index restaurant\_name\_index on dbcourse\_restaurant(name);  create index restaurant\_city\_index on dbcourse\_address(city);  create index restaurant\_food\_index on dbcourse\_food(name);  create index food\_price\_index on dbcourse\_food(price);  create index employee\_id\_index on dbcourse\_employee(id);  create index ingredient\_index on dbcourse\_ingredient(name);    create view DBCOURSE\_Delivery\_O as      select DBCOURSE\_Restaurant.name as restaurant\_name, city, DBCOURSE\_Food.type as type, DBCOURSE\_Food.name as food\_name, price      from (DBCOURSE\_Delivery natural join DBCOURSE\_Food,DBCOURSE\_Restaurant) natural join DBCOURSE\_Address      where DBCOURSE\_Food.restaurant\_name = DBCOURSE\_Restaurant.name and deliver = 'o'      group by DBCOURSE\_Food.name;    create view DBCOURSE\_Delivery\_X as      select DBCOURSE\_Restaurant.name as restaurant\_name, city, DBCOURSE\_Food.type as type, DBCOURSE\_Food.name as food\_name, price      from (DBCOURSE\_Delivery natural join DBCOURSE\_Food,DBCOURSE\_Restaurant) natural join DBCOURSE\_Address      where DBCOURSE\_Food.restaurant\_name = DBCOURSE\_Restaurant.name and deliver = 'x'      group by DBCOURSE\_Food.name;    create view DBCOURSE\_Order\_Ingredient as      select order\_number, DBCOURSE\_Order.restaurant\_name, DBCOURSE\_Order.food\_name, DBCOURSE\_Ingredient.name as ingredient     from DBCOURSE\_Food, DBCOURSE\_Order, DBCOURSE\_Ingredient     where DBCOURSE\_Food.restaurant\_name = DBCOURSE\_Order.restaurant\_name     and DBCOURSE\_Food.name = DBCOURSE\_Order.food\_name     and DBCOURSE\_Food.name = DBCOURSE\_Ingredient.food\_name;  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

-dropdb.sql

|  |  |
| --- | --- |
| use team1;  DROP VIEW dbcourse\_delivery\_o;  DROP VIEW dbcourse\_delivery\_x;  DROP VIEW dbcourse\_order\_ingredient;    DROP TABLE dbcourse\_Inventory;  DROP TABLE dbcourse\_Order;  DROP TABLE dbcourse\_Service;  DROP TABLE dbcourse\_Delivery;  DROP TABLE dbcourse\_Address;  DROP TABLE dbcourse\_Evaluation;  DROP TABLE dbcourse\_Employee;  DROP TABLE dbcourse\_Ingredient;  DROP TABLE dbcourse\_Food;  DROP TABLE dbcourse\_Restaurant; | [cs](http://colorscripter.com/info#e) |

1. **Java Codes**

|  |  |
| --- | --- |
| package team1;  import java.awt.\*;  import java.awt.event.\*;  import javax.swing.\*;  import javax.swing.table.\*;  import java.sql.\*;  import java.net.URL;    public class Test {      /\*\* 배소연\*/      //jdbc driver 이용      static final String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";      //database는 team1      static final String DB\_URL = "jdbc:mysql://localhost:3306/team1?useSSL=false&serverTimezone=UTC";      //user는 team1      static final String USER = "team1";      // password는 team1      static final String PASS = "team1";      //커넥션 설정을 위한 초기화      static  Connection conn = null;      //스테이트먼트 실행을 위한 초기화      static   Statement stmt = null;      static   Statement stmt2 = null;        public static void main(String[] args) {          try {              //Register JDBC driver              Class.forName(JDBC\_DRIVER);              //Open a connection              conn = DriverManager.getConnection(DB\_URL, USER, PASS);              //Execute a query              stmt = conn.createStatement();              stmt2 = conn.createStatement();          }catch(Exception e) {              e.printStackTrace();          }          /\*\* 박소연\*/          //frame 생성          REDSystem my = new REDSystem();      }      public static class REDSystem extends JFrame implements ActionListener, ItemListener {          //페이지별 카드          JPanel card, cardhome, card\_m1, card\_m2, card\_m3, card\_u1, card\_u2, card\_u3;          //페이지 별 버튼을 담는 패널          JPanel textpanel\_u, textpanel\_u2, radiobuttons, radiobuttons2, infopanel;          //홈페이지 배경을 그릴 main 패널과 나머지 페이지의 배경을 그릴 page 패널, user와 employee페이지의 테이블패널          JPanel main, page, tablepanel\_u, tablepanel\_emp;          //order, profit, salary up, salary down을 위한 패널          JPanel tablepanel\_order, tablepanel\_profit, bonuspanel, crisispanel;          //manager, user 모드 진입 버튼, 홈 버튼, 뒤로가기 버튼          JButton managerbutton, userbutton, homebutton, backbutton;          //각 페이지의 홈 버튼과 뒤로가기 버튼          JButton homebutton\_m1, homebutton\_m2, homebutton\_m3;          JButton homebutton\_u1, homebutton\_u2, homebutton\_u3;          JButton backbutton\_m1, backbutton\_m2, backbutton\_m3;          JButton backbutton\_u1, backbutton\_u2, backbutton\_u3;          //user모드에서 테이블 정렬 후 테이블 update 버튼과 next페이지로 넘어가는 버튼          JButton nextbutton\_u1, nextbutton\_u2, updatebutton;          //order table의 order를 처리하는 버튼과 user모드에서 delivery, nodelivery 선택 버튼          JButton takeorderbutton, deliverbutton, nodeliverbutton;          //manager모드에서 employee관리, orderprocessing 버튼, employee관리에서의 버튼          JButton employeebutton, orderprbutton, getinfobutton, enterinfobutton;          JButton givebutton\_up, givebutton\_down, firebutton, closebutton;          //다중 선택 버튼          Checkbox baby, pet, meat, seafood, dairy, fungus, vegetable, grain, fruit;          Checkbox family, date, friends, oriental, western, dessert;          Checkbox Seocho, Jongno, Seongdong, Songpa, Gangnam, Yongsan;          //하나만 선택해야 할 때 라디오 버튼          JRadioButton r5000, r10000, r15000, r20000, r25000, r30000;          JRadioButton update\_radio, delete\_radio, insert\_radio, park, nopark, nomatter;          JRadioButton r0, r2, r4, r6, r8, score\_radio, price\_radio, region\_radio;          JRadioButton remove\_radio, park\_radio, accompany\_radio, type\_radio, recommend\_radio;          //text입력 창          JTextField name, salary, plus\_salary, minus\_salary, delete\_salary;          //choice 상자          Choice c\_position, c\_res\_name;          //user2 페이지에서 쓰일 label          JLabel select, try\_again, success, restaurant\_name;          //user3 페이지에 출력되는 label          JLabel food\_type, food\_name, food\_price, restaurant\_num;          //m2 페이지에 안내 메세지 label          JLabel inform\_emp\_up\_1, inform\_emp\_down\_1,delivery\_cost\_label,delivery\_cost;          JLabel inform\_emp\_down\_errtable;          JLabel inform\_emp\_up\_done, inform\_emp\_down\_done;          JLabel inform\_emp\_error, inform\_emp\_down\_format;          JLabel inform\_emp\_insert, inform\_emp\_select;          //user3 페이지에 재료가 없을 때 출력되는 label, manager3 페이지에 쓰일 label          JLabel x, noingredient, complete, thisis, order, profit, empty;          //페이지와 버튼에 사용되는 이미지들          ImageIcon start, strawberry, pineapple, watermelon, kiwi, delivery, deliveryx;          ImageIcon manager1, manager2, manager3, user1, user2, user3, update, takeorder;          ImageIcon employee, cashier, orange, grape, home, back, next, close;          //gui 전체를 카드레이아웃으로 선언          CardLayout layout=new CardLayout();          //창 크기를 정하기 위해 스크린 사이즈를 불러오는 과정          Toolkit tk = getToolkit();          Dimension d = tk.getScreenSize();          //m2에서 테이블에서 받아온 정보와 새롭게 입력받은 정보          String emp\_ID, oldname, oldposition, oldresname, newname, newposition, newresname;          int oldsalary,newsalary;          //창 크기          int W = 9\*d.width/10;          int H = 9\*W/16;          //delivery, nodelivery를 구분하는 flag          int flag = 0;          //쿼리 실행 결과를 받아올 테이블모델          DefaultTableModel model, model2;          //테이블을 받아올 jtable          JTable table, table2;          //jtable을 반영할 스크롤팬          JScrollPane jsp, jsp2;            /\*\* 배소연\*/          //이미지를 크기에 맞게 바꾸는 method          ImageIcon changeImageSize(ImageIcon a, int m, int n) {              Image b = a.getImage();              b = b.getScaledInstance(W/m, H/n, Image.SCALE\_SMOOTH);              a = new ImageIcon(b);              return a;          }            /\*프레임창 생성자\*/          /\*\* 박소연\*/          public REDSystem(){              setTitle("Restaurant Management System");//타이틀을 설정한다.              setSize(W,H);//크기를 설정한다.              //card 패널에 cardlayout을 설정              card=new JPanel();              card.setLayout(layout);                /\*\* 배소연\*/              //이미지 불러오기              home = new ImageIcon(this.getClass().getClassLoader().getResource("homebutton.png"));              back = new ImageIcon(this.getClass().getClassLoader().getResource("backbutton.png"));              start = new ImageIcon(this.getClass().getClassLoader().getResource("start.png"));              next = new ImageIcon(this.getClass().getClassLoader().getResource("next.png"));              manager1 = new ImageIcon(this.getClass().getClassLoader().getResource("managerbutton.png"));              manager2 = new ImageIcon(this.getClass().getClassLoader().getResource("managerbutton2.png"));              manager3 = new ImageIcon(this.getClass().getClassLoader().getResource("managerbutton3.png"));              user1 = new ImageIcon(this.getClass().getClassLoader().getResource("userbutton.png"));              user2 = new ImageIcon(this.getClass().getClassLoader().getResource("userbutton2.png"));              user3 = new ImageIcon(this.getClass().getClassLoader().getResource("userbutton3.png"));              strawberry = new ImageIcon(this.getClass().getClassLoader().getResource("strawberry.png"));              pineapple = new ImageIcon(this.getClass().getClassLoader().getResource("pineapple.png"));              watermelon = new ImageIcon(this.getClass().getClassLoader().getResource("watermelon.png"));              kiwi = new ImageIcon(this.getClass().getClassLoader().getResource("kiwi.png"));              delivery=new ImageIcon(this.getClass().getClassLoader().getResource("delivery.png"));              deliveryx=new ImageIcon(this.getClass().getClassLoader().getResource("deliveryx.png"));              update = new ImageIcon(this.getClass().getClassLoader().getResource("updatebutton.png"));              takeorder = new ImageIcon(this.getClass().getClassLoader().getResource("takeorder.png"));              employee = new ImageIcon(this.getClass().getClassLoader().getResource("employee.png"));              cashier = new ImageIcon(this.getClass().getClassLoader().getResource("cashier.png"));              orange = new ImageIcon(this.getClass().getClassLoader().getResource("orange.png"));              grape = new ImageIcon(this.getClass().getClassLoader().getResource("grape.png"));              close = new ImageIcon(this.getClass().getClassLoader().getResource("closebutton.png"));                //화면 비율에 맞게 이미지 크기 조정              home = changeImageSize(home, 20, 12);              back = changeImageSize(back, 20, 12);              next = changeImageSize(next, 12, 12);              manager1 = changeImageSize(manager1, 5, 5);              manager2 = changeImageSize(manager2, 5, 5);              manager3 = changeImageSize(manager3, 5, 5);              user1 = changeImageSize(user1, 5, 5);              user2 = changeImageSize(user2, 5, 5);              user3 = changeImageSize(user3, 5, 5);              delivery = changeImageSize(delivery, 4, 2);              deliveryx = changeImageSize(deliveryx, 4, 2);              update = changeImageSize(update, 22, 13);              takeorder = changeImageSize(takeorder, 8, 14);              employee = changeImageSize(employee, 4, 2);              cashier = changeImageSize(cashier, 4, 2);              close = changeImageSize(close, 4, 3);                /\*\* 박소연\*/              //home 페이지 패널 생성              cardhome=new JPanel();              cardhome.setLayout(null);                /\*\* 배소연\*/              //home 페이지에 배경 그리기              main = new JPanel(){                  public void paint(Graphics g) {                      g.drawImage(start.getImage(), 0, 0, W, H, null);                  }              };              main.setBounds(0, 0, getWidth(), getHeight());                //homebutton 생성              homebutton = new JButton(home);              homebutton.setBounds(W/200, H/100, W/20, H/12);              homebutton.setBorderPainted(false);                //backbutton 생성              backbutton = new JButton(back);              backbutton.setBounds(12\*W/200, H/100, W/20, H/12);              backbutton.setBorderPainted(false);                //managerbutton 생성              managerbutton = new JButton(manager1);              managerbutton.setBorderPainted(false);  //경계선 없앰              managerbutton.setRolloverIcon(manager2);  //마우스 올리면 변하게              managerbutton.setPressedIcon(manager3);  //클릭할 때 변하게              managerbutton.setBounds(W/2, 5\*H/9, W/5, H/5);  //x, y, width, height              managerbutton.addActionListener(this);                //userbutton 생성              userbutton = new JButton(user1);              userbutton.setBorderPainted(false);              userbutton.setRolloverIcon(user2);              userbutton.setPressedIcon(user3);              userbutton.setBounds(3\*W/4, 5\*H/9, W/5, H/5);              userbutton.addActionListener(this);                //cardhome에 버튼과 패널을 붙인다              cardhome.add(homebutton);              cardhome.add(backbutton);              cardhome.add(managerbutton);              cardhome.add(userbutton);              cardhome.add(main);                //home카드를 card에 붙인다              card.add("home",cardhome);                /\*user 선택 시 - u1 page\*/              /\*\* 박소연\*/              //card layout에 붙일 새로운 패널 u1 생성              card\_u1 = new JPanel();              card\_u1.setLayout(null);                /\*\* 배소연\*/              //u1 card의 배경 및 위치 설정              page = new JPanel() {                  public void paint(Graphics g) {                      g.drawImage(pineapple.getImage(), 0, 0, W, H, null);                  }              };              page.setBounds(0, 0, getWidth(), getHeight());                //homebutton 생성              homebutton\_u1 = new JButton(home);              homebutton\_u1.setBounds(W/200, H/100, W/20, H/12);              homebutton\_u1.setBorderPainted(false);              homebutton\_u1.addActionListener(this);                //backbutton 생성              backbutton\_u1 = new JButton(back);              backbutton\_u1.setBounds(12\*W/200, H/100, W/20, H/12);              backbutton\_u1.setBorderPainted(false);              backbutton\_u1.addActionListener(this);                //deliverbutton 생성              deliverbutton = new JButton(delivery);              deliverbutton.setBounds(W/18, H/4, W/4, H/2);              deliverbutton.setBorderPainted(false);              deliverbutton.setContentAreaFilled(false);              deliverbutton.addActionListener(this);                //nodeliverbutton 생성              nodeliverbutton = new JButton(deliveryx);              nodeliverbutton.setBounds(W/3, H/4, W/4, H/2);              nodeliverbutton.setBorderPainted(false);              nodeliverbutton.setContentAreaFilled(false);              nodeliverbutton.addActionListener(this);                //card\_u1에 버튼과 패널을 붙인다              card\_u1.add(homebutton\_u1);              card\_u1.add(backbutton\_u1);              card\_u1.add(deliverbutton);              card\_u1.add(nodeliverbutton);              card\_u1.add(page);                //u1카드를 card에 붙인다              card.add("u1",card\_u1);                /\*u2 page \*/              /\*\* 박소연\*/              //card layout에 붙일 새로운 패널 u2 생성              card\_u2 = new JPanel();              card\_u2.setLayout(null);                /\*\* 배소연\*/              //u2 card의 배경 및 위치 설정              page = new JPanel() {                  public void paint(Graphics g) {                      g.drawImage(strawberry.getImage(), 0, 0, W, H, null);                  }              };              page.setBounds(0, 0, getWidth(), getHeight());                /\*\* 박소연\*/              //order에 전달 될 JLabel 초기화              restaurant\_name=new JLabel("");              food\_type=new JLabel("");              food\_name=new JLabel("");              food\_price=new JLabel("");              restaurant\_num=new JLabel("");                //radio button들을 담을 패널              JPanel scorepanel=new JPanel(new GridLayout(0,5));              JPanel pricepanel=new JPanel(new GridLayout(0,3));              JPanel regionpanel=new JPanel(new GridLayout(0,3));              JPanel removepanel = new JPanel(new GridLayout(0,4));              JPanel parkingpanel = new JPanel(new GridLayout(0,3));              JPanel accompanypanel = new JPanel(new GridLayout(0,2));              JPanel typepanel = new JPanel(new GridLayout(0,4));              JPanel recommendpanel = new JPanel(new GridLayout(0,4));                //검색 옵션 radio button              score\_radio=new JRadioButton("Score Option");              price\_radio=new JRadioButton("Price Option");              region\_radio=new JRadioButton("Region Option");              remove\_radio = new JRadioButton("I don't want this ingredient!");              park\_radio = new JRadioButton("Parking Option");              accompany\_radio = new JRadioButton("Accompany Option");              type\_radio = new JRadioButton("Food Type");              recommend\_radio = new JRadioButton("Recommend For");                //radio button들의 배경을 투명으로 설정하고 글씨체 설정              score\_radio.setOpaque(false);              price\_radio.setOpaque(false);              region\_radio.setOpaque(false);              remove\_radio.setOpaque(false);              park\_radio.setOpaque(false);              accompany\_radio.setOpaque(false);              type\_radio.setOpaque(false);              recommend\_radio.setOpaque(false);              score\_radio.setFont(new Font("Britannic Bold",Font.PLAIN,H/35));              region\_radio.setFont(new Font("Britannic Bold",Font.PLAIN,H/35));              price\_radio.setFont(new Font("Britannic Bold",Font.PLAIN,H/35));              remove\_radio.setFont(new Font("Britannic Bold",Font.PLAIN,H/35));              park\_radio.setFont(new Font("Britannic Bold",Font.PLAIN,H/35));              accompany\_radio.setFont(new Font("Britannic Bold",Font.PLAIN,H/35));              type\_radio.setFont(new Font("Britannic Bold",Font.PLAIN,H/35));              recommend\_radio.setFont(new Font("Britannic Bold",Font.PLAIN,H/35));                //옵션 radio button을 그룹으로 묶는다.              ButtonGroup detail=new ButtonGroup();              detail.add(price\_radio);detail.add(region\_radio);              detail.add(remove\_radio);detail.add(park\_radio);              detail.add(accompany\_radio);detail.add(type\_radio);              detail.add(recommend\_radio);detail.add(score\_radio);                //옵션을 담을 패널              JPanel score\_r=new JPanel();              JPanel region\_r=new JPanel();              JPanel price\_r=new JPanel();              JPanel remove\_r=new JPanel();              JPanel parking\_r=new JPanel();              JPanel accompany\_r=new JPanel();              JPanel type\_r=new JPanel();              JPanel recommend\_r=new JPanel();                //옵션을 담을 패널의 배경색을 흰색으로 설정              price\_r.setBackground(Color.white);region\_r.setBackground(Color.white);              remove\_r.setBackground(Color.white);parking\_r.setBackground(Color.white);              accompany\_r.setBackground(Color.white);type\_r.setBackground(Color.white);              recommend\_r.setBackground(Color.white);score\_r.setBackground(Color.white);                //score옵션의 radiobutton을 초기화              r0=new JRadioButton("ALL");              r2=new JRadioButton("over 2");              r4=new JRadioButton("over 4");              r6=new JRadioButton("over 6");              r8=new JRadioButton("over 8");                //패널에 score 옵션의 radiobutton들을 붙인다.              scorepanel.add(r0);scorepanel.add(r2);              scorepanel.add(r4);scorepanel.add(r6);scorepanel.add(r8);                //score 옵션의 radiobutton들을 그룹으로 묶는다.              ButtonGroup scoregroup=new ButtonGroup();              scoregroup.add(r0);scoregroup.add(r2);              scoregroup.add(r4);scoregroup.add(r6);scoregroup.add(r8);                //region 옵션의 checkbox을 초기화              Seocho= new Checkbox("Seocho");              Jongno= new Checkbox("Jongno");              Seongdong= new Checkbox("Seongdong");              Songpa= new Checkbox("Songpa");              Gangnam= new Checkbox("Gangnam");              Yongsan= new Checkbox("Yongsan");                //region 옵션의 radiobutton들을 패널에 붙인다.              regionpanel.add(Seocho);regionpanel.add(Jongno);              regionpanel.add(Seongdong);regionpanel.add(Songpa);              regionpanel.add(Gangnam);regionpanel.add(Yongsan);                //price 옵션의 radiobutton들을 초기화              r5000= new JRadioButton("~5000 won");              r10000= new JRadioButton("~10000 won");              r15000= new JRadioButton("~15000 won");              r20000= new JRadioButton("~20000 won");              r25000= new JRadioButton("~25000 won");              r30000= new JRadioButton("~30000 won");                //price 옵션의 radiobutton들을 패널에 붙인다.              pricepanel.add(r5000);pricepanel.add(r10000);              pricepanel.add(r15000);pricepanel.add(r20000);              pricepanel.add(r25000);pricepanel.add(r30000);                //price 옵션의 radiobutton들을 그룹으로 묶는다.              ButtonGroup pricegroup=new ButtonGroup();              pricegroup.add(r5000); pricegroup.add(r10000); pricegroup.add(r15000);              pricegroup.add(r20000); pricegroup.add(r25000); pricegroup.add(r30000);                //parking 옵션의 radiobutton들을 초기화              park = new JRadioButton("Parking");              nopark = new JRadioButton("NO Parking");              nomatter = new JRadioButton("NO matter");                //parking 옵션의 radiobutton들을 패널에 붙인다.              parkingpanel.add(park);parkingpanel.add(nopark);parkingpanel.add(nomatter);                //버튼에 actionlistener을 붙인다.              park.addActionListener(this); nopark.addActionListener(this);              nomatter.addActionListener(this);                //park 옵션의 radiobutton을 그룹으로 묶는다.              ButtonGroup park\_gr=new ButtonGroup();              park\_gr.add(park); park\_gr.add(nopark); park\_gr.add(nomatter);                //service 옵션의 checkbox를 초기화              baby = new Checkbox("with Baby");              pet = new Checkbox("with pet");                //패널에 service 옵션의 checkbox를 붙인다.              accompanypanel.add(baby); accompanypanel.add(pet);                //ingredient 옵션의 checkbox를 초기화              meat = new Checkbox("meat");              seafood = new Checkbox("seafood");              dairy = new Checkbox("dairy");              fungus = new Checkbox("fungus");              vegetable = new Checkbox("vegetable");              grain = new Checkbox("grain");              fruit = new Checkbox("fruit");                //ingredient 옵션의 radiobutton들을 패널에 붙인다.              removepanel.add(meat); removepanel.add(seafood); removepanel.add(dairy);              removepanel.add(fungus); removepanel.add(vegetable); removepanel.add(grain);              removepanel.add(fruit);                //recommend 옵션의 checkbox를 초기화              family = new Checkbox("family");              date = new Checkbox("date");              friends = new Checkbox("friends");              recommendpanel.add(family);recommendpanel.add(friends);recommendpanel.add(date);                //foodtype 옵션의 checkbox를 초기화              oriental = new Checkbox("oriental");              western = new Checkbox("western");              dessert = new Checkbox("dessert");                //패널에 food type 옵션의 checkbox를 붙인다.              typepanel.add(oriental);typepanel.add(western);typepanel.add(dessert);                //패널에 recommend 옵션의 checkbox와 옵션들을 붙인다.              score\_r.add(score\_radio);              price\_r.add(price\_radio);              region\_r.add(region\_radio);              remove\_r.add(remove\_radio);              parking\_r.add(park\_radio);              accompany\_r.add(accompany\_radio);              type\_r.add(type\_radio);              recommend\_r.add(recommend\_radio);                //radio button 묶음을 붙일 패널 - y축으로 붙이는 box layout으로 설정              radiobuttons = new JPanel();              radiobuttons2=new JPanel();              radiobuttons.setLayout(new BoxLayout(radiobuttons,BoxLayout.Y\_AXIS));              radiobuttons2.setLayout(new BoxLayout(radiobuttons2,BoxLayout.Y\_AXIS));                //radiobutton과 checkbox의 글씨 설정              Seocho.setFont(new Font("Britannic Bold",Font.PLAIN,H/40));              Jongno.setFont(new Font("Britannic Bold",Font.PLAIN,H/40));              Seongdong.setFont(new Font("Britannic Bold",Font.PLAIN,H/40));              Songpa.setFont(new Font("Britannic Bold",Font.PLAIN,H/40));              Gangnam.setFont(new Font("Britannic Bold",Font.PLAIN,H/40));              Yongsan.setFont(new Font("Britannic Bold",Font.PLAIN,H/40));              r5000.setFont(new Font("Courier",Font.PLAIN,H/40));              r10000.setFont(new Font("Courier",Font.PLAIN,H/40));              r15000.setFont(new Font("Courier",Font.PLAIN,H/40));              r20000.setFont(new Font("Courier",Font.PLAIN,H/40));              r25000.setFont(new Font("Courier",Font.PLAIN,H/40));              r30000.setFont(new Font("Courier",Font.PLAIN,H/40));              r0.setFont(new Font("Courier",Font.PLAIN,H/40));              r2.setFont(new Font("Courier",Font.PLAIN,H/40));              r4.setFont(new Font("Courier",Font.PLAIN,H/40));              r6.setFont(new Font("Courier",Font.PLAIN,H/40));              r8.setFont(new Font("Courier",Font.PLAIN,H/40));              meat.setFont(new Font("Courier",Font.PLAIN,H/40));              seafood.setFont(new Font("Courier",Font.PLAIN,H/40));              dairy.setFont(new Font("Courier",Font.PLAIN,H/40));              fungus.setFont(new Font("Courier",Font.PLAIN,H/40));              vegetable.setFont(new Font("Courier",Font.PLAIN,H/40));              grain.setFont(new Font("Courier",Font.PLAIN,H/40));              fruit.setFont(new Font("Courier",Font.PLAIN,H/40));              park.setFont(new Font("Courier",Font.PLAIN,H/40));              nopark.setFont(new Font("Courier",Font.PLAIN,H/40));              nomatter.setFont(new Font("Courier",Font.PLAIN,H/40));              pet.setFont(new Font("Courier",Font.PLAIN,H/40));              baby.setFont(new Font("Courier",Font.PLAIN,H/40));              family.setFont(new Font("Courier",Font.PLAIN,H/40));              date.setFont(new Font("Courier",Font.PLAIN,H/40));              friends.setFont(new Font("Courier",Font.PLAIN,H/40));              oriental.setFont(new Font("Courier",Font.PLAIN,H/40));              western.setFont(new Font("Courier",Font.PLAIN,H/40));              dessert.setFont(new Font("Courier",Font.PLAIN,H/40));                //패널에 라디오버튼과 체크박스가 들어있는 패널들을 붙인다.              textpanel\_u = new JPanel();              textpanel\_u2 = new JPanel();              radiobuttons.add(score\_r);              radiobuttons.add(scorepanel);              radiobuttons.add(region\_r);              radiobuttons.add(regionpanel);              radiobuttons.add(price\_r);              radiobuttons.add(pricepanel);              radiobuttons.add(remove\_r);              radiobuttons.add(removepanel);              radiobuttons.add(type\_r);              radiobuttons.add(typepanel);              radiobuttons.add(recommend\_r);              radiobuttons.add(recommendpanel);              radiobuttons2.add(accompany\_r);              radiobuttons2.add(accompanypanel);              radiobuttons2.add(parking\_r);              radiobuttons2.add(parkingpanel);                //라디오 버튼과 체크박스가 붙여져 있는 패널을 새로운 패널에 붙이고 배경 및 위치 설정              textpanel\_u.setLayout(new BoxLayout(textpanel\_u,BoxLayout.Y\_AXIS));              textpanel\_u.add(radiobuttons);              textpanel\_u2.add(radiobuttons2);              radiobuttons.setBackground(Color.white);              radiobuttons2.setBackground(Color.white);              textpanel\_u.setOpaque(false);              textpanel\_u2.setOpaque(false);              textpanel\_u.setBounds(W/50, H/10, 3\*W/7, 3\*H/4);              textpanel\_u2.setBounds(19\*W/40, 8\*H/12, W/3, H/5);                //not delivery일 때 선택하는 버튼들을 알리는 label              JLabel onlydelivery=new JLabel("Only for No Delivery");              onlydelivery.setFont(new Font("Britannic Bold",Font.PLAIN,H/28));              onlydelivery.setOpaque(false);              onlydelivery.setBounds(22\*W/40,15\*H/24,W/3,H/20);              onlydelivery.setBackground(Color.white);                /\*\* 배소연\*/              //table 위의 안내 message label              //선택 전              select = new JLabel("Select!", SwingConstants.CENTER);              select.setFont(new Font("Britannic Bold",Font.PLAIN,H/15));              select.setBounds(53\*W/100, H/70, W/5, H/15);                //에러가 났을 때              try\_again = new JLabel("Try Again!", SwingConstants.CENTER);              try\_again.setFont(new Font("Britannic Bold",Font.PLAIN,H/20));              try\_again.setBounds(53\*W/100, H/70, W/5, H/15);              try\_again.setVisible(false);                //선택이 성공적으로 완료되었을 때              success = new JLabel("Success!", SwingConstants.CENTER);              success.setFont(new Font("Britannic Bold",Font.PLAIN,H/18));              success.setBounds(53\*W/100, H/70, W/5, H/15);              success.setVisible(false);                //homebutton 생성              homebutton\_u2 = new JButton(home);              homebutton\_u2.setBounds(W/200, H/100, W/20, H/12);              homebutton\_u2.setBorderPainted(false);              homebutton\_u2.addActionListener(this);                //backbutton 생성              backbutton\_u2 = new JButton(back);              backbutton\_u2.setBounds(12\*W/200, H/100, W/20, H/12);              backbutton\_u2.setBorderPainted(false);              backbutton\_u2.addActionListener(this);                //nextbutton 생성              nextbutton\_u2 = new JButton(next);              nextbutton\_u2.setBounds(8\*W/9, H/100, W/12, H/12);              nextbutton\_u2.setBorderPainted(false);              nextbutton\_u2.setContentAreaFilled(false);              nextbutton\_u2.addActionListener(this);                //updatebutton 생성              updatebutton = new JButton(update);              updatebutton.setBounds(15\*W/18, H/100, W/22, H/13);              updatebutton.setBorderPainted(false);              updatebutton.setContentAreaFilled(false);              updatebutton.addActionListener(this);                //table 패널 생성              tablepanel\_u = new JPanel();                //card\_u2에 버튼과 패널을 붙인다              card\_u2.add(onlydelivery);              card\_u2.add(textpanel\_u);              card\_u2.add(textpanel\_u2);              card\_u2.add(homebutton\_u2);              card\_u2.add(backbutton\_u2);              card\_u2.add(nextbutton\_u2);              card\_u2.add(updatebutton);              card\_u2.add(select);              card\_u2.add(try\_again);              card\_u2.add(success);              card\_u2.add(tablepanel\_u);              card\_u2.add(page);                //card\_u2 카드를 card에 붙인다              card.add("u2", card\_u2);                /\*u3 화면\*/              /\*\* 박소연\*/              //card layout에 붙일 새로운 패널 u3 생성              card\_u3 = new JPanel();              card\_u3.setLayout(null);                /\*\* 배소연\*/              //u3의 배경 및 위치 설정              page = new JPanel() {                  public void paint(Graphics g) {                      g.drawImage(kiwi.getImage(), 0, 0, W, H, null);                  }              };              page.setBounds(0, 0, getWidth(), getHeight());                /\*\* 박소연\*/              //order 확인 label과 패널              JPanel orderpanel=new JPanel();              JLabel column0=new JLabel("RESTAURANT NAME: ");              column0.setFont(new Font("Britannic Bold",Font.BOLD,H/35));              JLabel column1=new JLabel("FOOD TYPE: ");              column1.setFont(new Font("Britannic Bold",Font.BOLD,H/35));              JLabel column2=new JLabel("FOOD NAME: ");              column2.setFont(new Font("Britannic Bold",Font.BOLD,H/35));              JLabel column3=new JLabel("FOOD PRICE: ");              column3.setFont(new Font("Britannic Bold",Font.BOLD,H/35));              JLabel column4=new JLabel("RESTAURANT NUMBER: ");              column4.setFont(new Font("Britannic Bold",Font.BOLD,H/35));              delivery\_cost\_label=new JLabel("");              delivery\_cost=new JLabel("");              delivery\_cost\_label.setFont(new Font("Britannic Bold",Font.BOLD,H/35));                //label을 패널에 붙인다.              orderpanel.add(column0);orderpanel.add(restaurant\_name);              orderpanel.add(column1);orderpanel.add(food\_type);              orderpanel.add(column2);orderpanel.add(food\_name);              orderpanel.add(column3);orderpanel.add(food\_price);              orderpanel.add(column4);orderpanel.add(restaurant\_num);              orderpanel.add(delivery\_cost\_label); orderpanel.add(delivery\_cost);              orderpanel.setLayout(new BoxLayout(orderpanel,BoxLayout.Y\_AXIS));              orderpanel.setBounds(W/13,13\*H/40,W/3,H/2);              orderpanel.setOpaque(false);                /\*\* 배소연\*/              //재료가 없을 때 뜨는 X label 생성              x = new JLabel("X");              x.setFont(new Font("Courier",Font.BOLD,2\*H/3));              x.setForeground(Color.RED);              x.setBounds(W/13,H/4,W/3,2\*H/3);              x.setVisible(false);                //sorry, no ingredient label 생성              noingredient = new JLabel("Sorry, No Ingredient");              noingredient.setFont(new Font("Britannic Bold",Font.BOLD,H/10));              noingredient.setBounds(2\*W/13,H/20,W,H/10);              noingredient.setVisible(false);                //complete order label 생성              complete = new JLabel("Complete Order!");              complete.setFont(new Font("Britannic Bold",Font.BOLD,H/10));              complete.setBounds(2\*W/13,H/20,W,H/10);                //homebutton 생성              homebutton\_u3 = new JButton(home);              homebutton\_u3.setBounds(W/200, H/100, W/20, H/12);              homebutton\_u3.setBorderPainted(false);              homebutton\_u3.addActionListener(this);                //backbutton 생성              backbutton\_u3 = new JButton(back);              backbutton\_u3.setBounds(12\*W/200, H/100, W/20, H/12);              backbutton\_u3.setBorderPainted(false);              backbutton\_u3.addActionListener(this);                //card\_u3에 버튼과 패널을 붙인다              card\_u3.add(homebutton\_u3);              card\_u3.add(backbutton\_u3);              card\_u3.add(orderpanel);              card\_u3.add(noingredient);              card\_u3.add(complete);              card\_u3.add(x);              card\_u3.add(page);                //card\_u3 카드를 card에 붙인다              card.add("u3", card\_u3);                /\*manager 선택 시 - m1 page\*/              /\*\* 배소연\*/              //card layout에 붙일 새로운 패널 m1 생성              card\_m1 = new JPanel();              card\_m1.setLayout(null);                //m1 card의 배경 및 위치 설정              page = new JPanel() {                  public void paint(Graphics g) {                      g.drawImage(watermelon.getImage(), 0, 0, W, H, null);                  }              };              page.setBounds(0, 0, getWidth(), getHeight());                //homebutton 생성              homebutton\_m1 = new JButton(home);              homebutton\_m1.setBounds(W/200, H/100, W/20, H/12);              homebutton\_m1.setBorderPainted(false);              homebutton\_m1.addActionListener(this);                //backbutton 생성              backbutton\_m1 = new JButton(back);              backbutton\_m1.setBounds(12\*W/200, H/100, W/20, H/12);              backbutton\_m1.setBorderPainted(false);              backbutton\_m1.addActionListener(this);                //employee management 버튼 생성              employeebutton = new JButton(employee);              employeebutton.setBounds(W/18, H/4, W/4, H/2);              employeebutton.setBorderPainted(false);              employeebutton.setContentAreaFilled(false);              employeebutton.addActionListener(this);                //order processing 버튼 생성              orderprbutton = new JButton(cashier);              orderprbutton.setBounds(W/3, H/4, W/4, H/2);              orderprbutton.setBorderPainted(false);              orderprbutton.setContentAreaFilled(false);              orderprbutton.addActionListener(this);                //card\_m1에 버튼과 패널을 붙인다              card\_m1.add(homebutton\_m1);              card\_m1.add(backbutton\_m1);              card\_m1.add(employeebutton);              card\_m1.add(orderprbutton);              card\_m1.add(page);                //card\_m1 카드를 card에 붙인다              card.add("m1", card\_m1);                /\*m2 page\*/              /\*\* 배소연 \*/              //card layout에 붙일 새로운 패널 m2 생성              card\_m2 = new JPanel();              card\_m2.setLayout(null);                //m2 card의 배경 및 위치 설정              page = new JPanel() {                  public void paint(Graphics g) {                      g.drawImage(orange.getImage(), 0, 0, W, H, null);                  }              };              page.setBounds(0, 0, getWidth(), getHeight());                //homebutton 생성              homebutton\_m2 = new JButton(home);              homebutton\_m2.setBounds(W/200, H/100, W/20, H/12);              homebutton\_m2.setBorderPainted(false);              homebutton\_m2.addActionListener(this);                //backbutton 생성              backbutton\_m2 = new JButton(back);              backbutton\_m2.setBounds(12\*W/200, H/100, W/20, H/12);              backbutton\_m2.setBorderPainted(false);              backbutton\_m2.addActionListener(this);                //employee 테이블을 그릴 패널 생성              tablepanel\_emp = new JPanel();                /\*\* 박소연\*/              //각 패널의 역할을 알리는 label과 그 label의 글씨 설정              JLabel bonuslabel=new JLabel("Change Salary!");              bonuslabel.setFont(new Font("Britannic Bold",Font.PLAIN,H/30));              JLabel crisislabel\_up=new JLabel("Fire employee!");              crisislabel\_up.setFont(new Font("Britannic Bold",Font.PLAIN,H/25));              JLabel crisislabel\_down=new JLabel("salary over ");              crisislabel\_down.setFont(new Font("Britannic Bold",Font.PLAIN,H/25));              JLabel inputlabel\_up=new JLabel("Input :");              inputlabel\_up.setFont(new Font("Britannic Bold",Font.PLAIN,H/35));              JLabel inputlabel\_down=new JLabel("Input :");              inputlabel\_down.setFont(new Font("Britannic Bold",Font.PLAIN,H/35));              JLabel inclabel\_p=new JLabel("increase payment");              inclabel\_p.setFont(new Font("Britannic Bold",Font.PLAIN,H/40));              JLabel declabel\_p=new JLabel("decrease payment");              declabel\_p.setFont(new Font("Britannic Bold",Font.PLAIN,H/40));                //안내메세지 label 설정              inform\_emp\_up\_1=new JLabel("Check & Change");              inform\_emp\_up\_1.setFont(new Font("Britannic Bold",Font.PLAIN,H/18));              inform\_emp\_up\_1.setBounds(W/40, 21\*H/100, W/3, H/18);              inform\_emp\_up\_1.setVisible(true);                inform\_emp\_down\_1=new JLabel("Employee List!");              inform\_emp\_down\_1.setFont(new Font("Britannic Bold",Font.PLAIN,H/15));              inform\_emp\_down\_1.setBounds(W/50, 7\*H/25, W/4, H/15);              inform\_emp\_down\_1.setVisible(true);                inform\_emp\_up\_done=new JLabel("Done!");              inform\_emp\_up\_done.setFont(new Font("Britannic Bold",Font.PLAIN,H/10));              inform\_emp\_up\_done.setBounds(W/15, H/6, W/4, H/10);              inform\_emp\_up\_done.setVisible(false);                inform\_emp\_down\_done=new JLabel("Identify Table!");              inform\_emp\_down\_done.setFont(new Font("Britannic Bold",Font.PLAIN,H/15));              inform\_emp\_down\_done.setBounds(W/40, 7\*H/25, W/4, H/15);              inform\_emp\_down\_done.setVisible(false);                inform\_emp\_error=new JLabel("Error!");              inform\_emp\_error.setFont(new Font("Britannic Bold",Font.PLAIN,H/10));              inform\_emp\_error.setBounds(W/15, H/6, W/4, H/10);              inform\_emp\_error.setVisible(false);                inform\_emp\_down\_format=new JLabel("Input Properly!");              inform\_emp\_down\_format.setFont(new Font("Britannic Bold",Font.PLAIN,H/16));              inform\_emp\_down\_format.setBounds(W/45, 7\*H/25, W/4, H/16);              inform\_emp\_down\_format.setVisible(false);                inform\_emp\_down\_errtable=new JLabel("Select Table!");              inform\_emp\_down\_errtable.setFont(new Font("Britannic Bold",Font.PLAIN,H/15));              inform\_emp\_down\_errtable.setBounds(W/35, 7\*H/25, W/4, H/15);              inform\_emp\_down\_errtable.setVisible(false);                /\*\* 배소연\*/              inform\_emp\_select = new JLabel("Select Button!");              inform\_emp\_select.setFont(new Font("Britannic Bold",Font.PLAIN,H/15));              inform\_emp\_select.setBounds(W/45, 7\*H/25, W/4, H/15);              inform\_emp\_select.setVisible(false);                inform\_emp\_insert = new JLabel("Insert!");              inform\_emp\_insert.setFont(new Font("Britannic Bold",Font.PLAIN,H/10));              inform\_emp\_insert.setBounds(W/17, 7\*H/25, W/4, H/10);              inform\_emp\_insert.setVisible(false);                  //table로부터 받아오거나 table을 반영시킬 정보를 담을 패널              infopanel=new JPanel();              infopanel.setLayout(new BoxLayout(infopanel,BoxLayout.Y\_AXIS));              bonuspanel=new JPanel();crisispanel=new JPanel();              bonuspanel.setLayout(new BoxLayout(bonuspanel,BoxLayout.Y\_AXIS));              crisispanel.setLayout(new BoxLayout(crisispanel,BoxLayout.Y\_AXIS));              JPanel info\_radiopanel=new JPanel(new GridLayout(0,4));              JPanel info\_tablepanel=new JPanel(new GridLayout(0,5));              JPanel pluspanel=new JPanel(new GridLayout(0,3));              JPanel minuspanel=new JPanel(new GridLayout(0,3));              JPanel dismisspanel=new JPanel(new GridLayout(0,3));                //update, delete, insert 선택할 버튼과 그 글씨체를 설정하고 그룹으로 묶는다.              ButtonGroup infogroup=new ButtonGroup();              update\_radio=new JRadioButton("update"); update\_radio.addActionListener(this);              delete\_radio=new JRadioButton("delete"); delete\_radio.addActionListener(this);              insert\_radio=new JRadioButton("insert"); insert\_radio.addActionListener(this);              update\_radio.setFont(new Font("Britannic Bold",Font.PLAIN,H/30));              delete\_radio.setFont(new Font("Britannic Bold",Font.PLAIN,H/30));              insert\_radio.setFont(new Font("Britannic Bold",Font.PLAIN,H/30));              infogroup.add(update\_radio); infogroup.add(delete\_radio); infogroup.add(insert\_radio);                //테이블로부터 정보를 받아올 버튼              getinfobutton=new JButton("GET INFO");              getinfobutton.setFont(new Font("Britannic Bold",Font.PLAIN,H/30));              getinfobutton.addActionListener(this);                //테이블에 정보를 반영시킬 버튼              enterinfobutton=new JButton("ENTER");              enterinfobutton.setFont(new Font("Britannic Bold",Font.PLAIN,H/30));              enterinfobutton.addActionListener(this);              givebutton\_up=new JButton("% UP");              givebutton\_up.setFont(new Font("Britannic Bold",Font.PLAIN,H/33));              givebutton\_up.addActionListener(this);              givebutton\_down=new JButton("% DOWN");              givebutton\_down.setFont(new Font("Britannic Bold",Font.PLAIN,H/33));              givebutton\_down.addActionListener(this);              firebutton=new JButton("BYE!");              firebutton.setFont(new Font("Britannic Bold",Font.PLAIN,H/30));              firebutton.addActionListener(this);                //information을 바꿀 옵션 radio button을 패널에 붙인다.              info\_radiopanel.add(update\_radio);info\_radiopanel.add(delete\_radio);              info\_radiopanel.add(insert\_radio);info\_radiopanel.add(getinfobutton);                //개인의 정보를 보여줄 textfield와 choice 생성              name=new JTextField(W/9); name.addActionListener(this);              name.setFont(new Font("Courier",Font.PLAIN,H/33));              salary=new JTextField(W/9); salary.addActionListener(this);              salary.setFont(new Font("Courier",Font.PLAIN,H/33));              plus\_salary=new JTextField(W/4); plus\_salary.addActionListener(this);              plus\_salary.setFont(new Font("Courier",Font.PLAIN,H/30));              minus\_salary=new JTextField(W/4); minus\_salary.addActionListener(this);              minus\_salary.setFont(new Font("Courier",Font.PLAIN,H/30));              delete\_salary=new JTextField(W/8); delete\_salary.addActionListener(this);              delete\_salary.setFont(new Font("Courier",Font.PLAIN,H/30));              c\_position=new Choice(); c\_position.addItemListener(this);              c\_res\_name=new Choice(); c\_res\_name.addItemListener(this);              c\_position.add("chef");c\_position.add("cook");              c\_position.add("manager");c\_position.add("server");              c\_res\_name.add("Nolboo");c\_res\_name.add("SchoolFood");              c\_res\_name.add("Abiko");c\_res\_name.add("Yummy Sushi");              c\_res\_name.add("Outback");c\_res\_name.add("VIPS");              c\_res\_name.add("Ewhasung");c\_res\_name.add("BillyAngel");              c\_res\_name.add("Sulbing");c\_res\_name.add("KFC");              c\_res\_name.add("Shake Shack");              c\_position.setFont(new Font("Courier",Font.PLAIN,H/35));              c\_res\_name.setFont(new Font("Courier",Font.PLAIN,H/35));                //패널에 textfield, JButton, 그리고 choice를 붙인다.              info\_tablepanel.add(name);              info\_tablepanel.add(c\_res\_name);              info\_tablepanel.add(c\_position);              info\_tablepanel.add(salary);              info\_tablepanel.add(enterinfobutton);              pluspanel.add(inputlabel\_up);              pluspanel.add(plus\_salary);pluspanel.add(givebutton\_up);              minuspanel.add(inputlabel\_down);              minuspanel.add(minus\_salary);minuspanel.add(givebutton\_down);              bonuspanel.add(bonuslabel);              bonuspanel.add(pluspanel);              bonuspanel.add(minuspanel);              dismisspanel.add(crisislabel\_down);              dismisspanel.add(delete\_salary);dismisspanel.add(firebutton);              crisispanel.add(crisislabel\_up);crisispanel.add(dismisspanel);              infopanel.add(info\_radiopanel);infopanel.add(info\_tablepanel);                //panel의 위치 설정              infopanel.setBounds(4\*W/15, 13\*H/20, 2\*W/3, H/11);              bonuspanel.setBounds(34\*W/50, 19\*H/25, W/4, H/7);              crisispanel.setBounds(4\*W/15, 19\*H/25, 2\*W/5, H/10);                //카드에 안내메세지 label들을 붙이고 현재 보여줄 메세지만 true로 설정              card\_m2.add(inform\_emp\_up\_1);              card\_m2.add(inform\_emp\_down\_1);              card\_m2.add(inform\_emp\_error);              card\_m2.add(inform\_emp\_down\_format);              card\_m2.add(inform\_emp\_down\_errtable);              card\_m2.add(inform\_emp\_up\_done);              card\_m2.add(inform\_emp\_down\_done);              card\_m2.add(inform\_emp\_select);              card\_m2.add(inform\_emp\_insert);                //card에 앞서 만든 패널들을 붙인다.              card\_m2.add(crisispanel);              card\_m2.add(infopanel);              card\_m2.add(bonuspanel);              card\_m2.add(homebutton\_m2);              card\_m2.add(backbutton\_m2);              card\_m2.add(tablepanel\_emp);              card\_m2.add(page);              card.add("m2", card\_m2);                /\*m3 page\*/              /\*\* 박소연\*/              //card layout에 붙일 새로운 패널 m3 생성              card\_m3 = new JPanel();              card\_m3.setLayout(null);                /\*\* 배소연\*/              //m3 card의 배경 및 위치 설정              page = new JPanel() {                  public void paint(Graphics g) {                      g.drawImage(grape.getImage(), 0, 0, W, H, null);                  }              };              page.setBounds(0, 0, getWidth(), getHeight());                //homebutton 생성              homebutton\_m3 = new JButton(home);              homebutton\_m3.setBounds(W/200, H/100, W/20, H/12);              homebutton\_m3.setBorderPainted(false);              homebutton\_m3.addActionListener(this);                //backbutton 생성              backbutton\_m3 = new JButton(back);              backbutton\_m3.setBounds(12\*W/200, H/100, W/20, H/12);              backbutton\_m3.setBorderPainted(false);              backbutton\_m3.addActionListener(this);                //order table의 order를 차례대로 처리하는 takeorderbutton 생성              takeorderbutton = new JButton(takeorder);              takeorderbutton.setBounds(23\*W/50, 9\*H/20, W/8, H/14);              takeorderbutton.setBorderPainted(false);              takeorderbutton.setContentAreaFilled(false);              takeorderbutton.addActionListener(this);                //영업종료시 daily wage를 지급하고 close하는 closebutton 생성              closebutton = new JButton(close);              closebutton.setBounds(2\*W/3, 23\*H/40, W/4, H/3);              closebutton.setBorderPainted(false);              closebutton.setContentAreaFilled(false);              closebutton.addActionListener(this);                //order table을 위한 패널              tablepanel\_order = new JPanel();                //restaurant 별 profit을 보여주는 table을 위한 패널              tablepanel\_profit = new JPanel();                //thisis label 생성              thisis = new JLabel("This is");              thisis.setFont(new Font("Britannic Bold",Font.PLAIN,H/10));              thisis.setBounds(39\*W/100, 29\*H/50, W/5, H/10);                //order list label 생성              order = new JLabel("Order List!");              order.setFont(new Font("Britannic Bold",Font.PLAIN,H/10));              order.setBounds(17\*W/50, 69\*H/100, W/3, H/10);                //today's profit label 생성              profit = new JLabel("Today's profit!");              profit.setFont(new Font("Britannic Bold",Font.PLAIN,H/11));              profit.setBounds(16\*W/50, 69\*H/100, 2\*W/5, H/11);              profit.setVisible(false);                //order table의 order가 아직 남았는데 closebutton을 누른 경우 뜨는 label              empty = new JLabel("Empty the list!");              empty.setFont(new Font("Britannic Bold",Font.PLAIN,H/10));              empty.setBounds(29\*W/100, 64\*H/100, 2\*W/5, H/10);              empty.setVisible(false);                //card\_m3에 버튼과 패널을 붙인다              card\_m3.add(homebutton\_m3);              card\_m3.add(backbutton\_m3);              card\_m3.add(takeorderbutton);              card\_m3.add(closebutton);              card\_m3.add(tablepanel\_order);              card\_m3.add(tablepanel\_profit);              card\_m3.add(thisis);              card\_m3.add(order);              card\_m3.add(profit);              card\_m3.add(empty);              card\_m3.add(page);                //card\_m3 카드를 card에 붙인다              card.add("m3", card\_m3);                //전체 card를 add              add(card);              //x를 누르면 프로그램과 함께 종료하도록 설정              setDefaultCloseOperation(EXIT\_ON\_CLOSE);              setVisible(true); //visible하게 설정한다.          }            /\*버튼 효과 처리 method\*/          public void actionPerformed(ActionEvent e) {              /\*\* 배소연 \*/              //home 페이지로 가는 경우              if(e.getSource() == homebutton\_u1 || e.getSource() == homebutton\_u2 || e.getSource() == homebutton\_u3 || e.getSource() == homebutton\_m1 || e.getSource() == homebutton\_m2 || e.getSource() == homebutton\_m3 || e.getSource() == backbutton\_u1 || e.getSource() == backbutton\_m1) {                  layout.show(card, "home");              }              //m1 페이지로 가는 경우              else if(e.getSource()==managerbutton || e.getSource() == backbutton\_m2 || e.getSource() == backbutton\_m3){                  layout.show(card, "m1");              }              //u1 페이지로 가는 경우              else if(e.getSource() == userbutton || e.getSource() == backbutton\_u2){                  layout.show(card, "u1");              }              //u2 페이지로 가는 경우              else if(e.getSource() == backbutton\_u3) {                  layout.show(card, "u2");              }              /\*\* 박소연\*/              else if(e.getSource() == employeebutton) {                  //employee 버튼을 누르면 안내메세지가 보이도록 설정                  inform\_emp\_up\_1.setVisible(true);                  inform\_emp\_down\_1.setVisible(true);                  inform\_emp\_up\_done.setVisible(false);                  inform\_emp\_down\_done.setVisible(false);                  inform\_emp\_down\_errtable.setVisible(false);                  inform\_emp\_error.setVisible(false);                  inform\_emp\_down\_format.setVisible(false);                  inform\_emp\_select.setVisible(false);                  inform\_emp\_insert.setVisible(false);                    //table의 column name을 초기화                  String[] colNames = {"ID", "name", "restaurant name", "position","salary(daily)", };                  //table을 update하도록 비운다.                  model = new DefaultTableModel(colNames, 0) {                      public Class<?> getColumnClass(int column) {                          if (column == 4) {                              return Integer.class;                          }                          return super.getColumnClass(column);                      }                  };                  try {                      //DBCOURSE\_Employee로부터 정보를 가져온다.                      ResultSet rs = stmt.executeQuery("select \* from DBCOURSE\_Employee");                      while(rs.next()) {                          Object data[] = {rs.getString(1), rs.getString(2), rs.getString(3), rs.getString(4), rs.getInt(5)};                          model.addRow(data);                      }                  } catch (SQLException e1) {                      e1.printStackTrace();                  }                  //employee table을 보여준다                  showTable\_emp();                  //m2 card를 보여준다                  layout.show(card, "m2");              }              /\*\* 배소연\*/              //manager모드의 order processing 버튼을 누른 경우              else if(e.getSource() == orderprbutton) {                  //order table에 들어갈 attribute array                  String[] colNames = {"number", "restaurant\_name", "food\_name", "price"};                  //DefaultTableModel 생성 - int 형의 column의 정렬을 위해 override                  model = new DefaultTableModel(colNames, 0) {                      public Class<?> getColumnClass(int column) {                          if (column == 4) {                              return Integer.class;                          }                          return super.getColumnClass(column);                      }                  };                  //profit을 보여주는 table에 들어갈 attribute array                  String[] colNames2 = {"restaurant\_name", "profit"};                  //DefaultTableModel 생성 - int 형의 column의 정렬을 위해 override                  model2 = new DefaultTableModel(colNames2, 0) {                      public Class<?> getColumnClass(int column) {                          if (column == 1) {                              return Integer.class;                          }                          return super.getColumnClass(column);                      }                  };                  try {                      //order table의 모든 tuple 받아오기                      ResultSet rs = stmt.executeQuery("select \* from DBCOURSE\_Order");                      //한 tuple씩 받아온다                      while(rs.next()) {                          Object data[] = {rs.getInt(1), rs.getString(2), rs.getString(3), rs.getInt(4)};                          model.addRow(data);                      }                      //restaurant table에서 모든 restaurant\_name과 profit 받아오기                      rs = stmt.executeQuery("select name as restaurant\_name, profit from DBCOURSE\_Restaurant");                      //한 줄씩 읽는다                      while(rs.next()) {                          Object data[] = {rs.getString(1), rs.getInt(2)};                          model2.addRow(data);                      }                  } catch (SQLException e1) {                      e1.printStackTrace();                  }                  //order table을 보여준다                  showTable\_order();                  //profit을 보여주는 table을 보여준다                  showTable\_profit();                  //m3 card를 보여준다                  layout.show(card, "m3");              }              /\*\* 배소연\*/              //u1페이지에서 delivery를 선택한 경우              else if(e.getSource() == deliverbutton) {                  flag = 0; //delivery인 경우 flag를 0으로 설정                  //delivery\_O view를 보여주는 table에 들어갈 attribute array                  String[] colNames = {"restaurant", "city", "type", "food", "price"};                  //DefaultTableModel 생성 - int 형의 column의 정렬을 위해 override                  model = new DefaultTableModel(colNames, 0) {                      public Class<?> getColumnClass(int column) {                          if (column == 4) {                              return Integer.class;                          }                          return super.getColumnClass(column);                      }                  };                  try {                      //delivery\_O view의 모든 tuple을 받아오기                      ResultSet rs = stmt.executeQuery("select \* from DBCOURSE\_Delivery\_O");                      //한 tuple씩 받아온다                      while(rs.next()) {                          Object data[] = {rs.getString(1), rs.getString(2), rs.getString(3), rs.getString(4), rs.getInt(5)};                          model.addRow(data);                      }                  } catch (SQLException e1) {                      e1.printStackTrace();                  }                  //user모드에서의 테이블을 보여준다                  showTable\_user();                  //u2 페이지를 보여준다                  layout.show(card, "u2");              }              /\*\* 배소연\*/              //u1페이지에서 No delivery를 선택한 경우              else if(e.getSource() == nodeliverbutton) {                  flag = 1; //No delivery인 경우 flag를 1로 설정                  //delivery\_X view를 보여주는 table에 들어갈 attribute array                  String[] colNames = {"restaurant", "city", "type", "food", "price"};                  //DefaultTableModel 생성 - int 형의 column의 정렬을 위해 override                  model = new DefaultTableModel(colNames, 0) {                      public Class<?> getColumnClass(int column) {                          if (column == 4) {                              return Integer.class;                          }                          return super.getColumnClass(column);                      }                  };                  try {                      //delivery\_X view의 모든 tuple을 받아오기                      ResultSet rs = stmt.executeQuery("select \* from DBCOURSE\_Delivery\_X");                      //한 tuple씩 받아온다                      while(rs.next()) {                          Object data[] = {rs.getString(1), rs.getString(2), rs.getString(3), rs.getString(4), rs.getInt(5)};                          model.addRow(data);                      }                  } catch (SQLException e1) {                      e1.printStackTrace();                  }                  //user모드에서의 테이블을 보여준다                  showTable\_user();                  //u2 페이지를 보여준다                  layout.show(card, "u2");              }              /\*\* 배소연\*/              //u2페이지에서 updatebutton을 누른 경우              else if(e.getSource() == updatebutton) {                  String S = ""; //sql문이 들어갈 String S                  //table에 들어갈 attribute array                  String[] colNames = {"restaurant", "city", "type", "food", "price"};                    model.setRowCount(0); //테이블 갱신을 위해 초기화                    /\*\* 배소연, 김해리, 신유진 \*/                  //score 기준을 선택했을 때                  if(score\_radio.isSelected()){                      //delivery인 경우                      if(flag == 0) S = "select restaurant\_name, city, type, food\_name, price " +                              "from (select restaurant\_name, city, type, food\_name, price, avg(score) as avg " +                              "from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Evaluation " +                              "group by food\_name) as A where avg";                      //No delivery인 경우                      else S = "select restaurant\_name, city, type, food\_name, price " +                              "from (select restaurant\_name, city, type, food\_name, price, avg(score) as avg " +                              "from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Evaluation " +                              "group by food\_name) as A where avg";                      //제대로 선택한 경우 success label 보여준다                      select.setVisible(false);                      try\_again.setVisible(false);                      success.setVisible(true);                      /\*\* 박소연\*/                      //all을 선택한 경우                      if(r0.isSelected()){                          S+=">=0";                      }                      //over2를 선택한 경우                      else if(r2.isSelected()){                          S+=">=2";                      }                      //over4를 선택한 경우                      else if(r4.isSelected()){                          S+=">=4";                      }                      //over6을 선택한 경우                      else if(r6.isSelected()){                          S+=">=6";                      }                      //over8을 선택한 경우                      else if(r8.isSelected()){                          S+=">=8";                      }                      /\*\* 배소연\*/                      //아무것도 선택하지 않고 updatebutton을 누른 경우                      else{                          S+="<0";                          //try again label을 보여준다                          select.setVisible(false);                          success.setVisible(false);                          try\_again.setVisible(true);                      }                  }                    /\*\* 박소연\*/                  //region 기준을 선택했을 때                  else if(region\_radio.isSelected()) {                      /\*\* sql - 신유진\*/                      //delivery인 경우                      if(flag == 0) S = "select restaurant\_name, city, type, food\_name, price " +                              "from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Address";                      //No delivery인 경우                      else S = "select restaurant\_name, city, type, food\_name, price " +                              "from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Address";                      //다중 선택 체크박스 배열                      Checkbox[] check = {Seocho, Jongno, Seongdong, Songpa, Gangnam, Yongsan};                      int count = 0;                      //체크한 도시를 추가하는 for문                      for(int i=0; i<check.length; i++) {                          if(check[i].getState() == true) {                              count++;                              if(count == 1) S += " where city = '" + check[i].getLabel() + "'";                              else S += " or city = '" + check[i].getLabel() + "'";                          }                      }                      /\*\* 배소연\*/                      //아무 도시도 체크하지 않은 경우                      if(count == 0) {                          S += " where city = ''";                          select.setVisible(false);                          success.setVisible(false);                          try\_again.setVisible(true);                      }                      //제대로 체크한 경우 success label을 보여준다                      else {                          select.setVisible(false);                          try\_again.setVisible(false);                          success.setVisible(true);                      }                  }                  /\*\* 박소연\*/                  //price 기준을 선택했을 때                  else if(price\_radio.isSelected()) {                      //price 조건을 제대로 선택한 경우 success label을 보여준다                      select.setVisible(false);                      try\_again.setVisible(false);                      success.setVisible(true);                      //delivery인 경우                      if(flag == 0) S = "select \* from DBCOURSE\_Delivery\_O";                      //No delivery인 경우                      else S = "select \* from DBCOURSE\_Delivery\_X";                      //선택한 조건에 따라 where문 추가                      if(r5000.isSelected()) S += " where price < 5000";                      else if(r10000.isSelected()) S += " where price < 10000";                      else if(r15000.isSelected()) S += " where price < 15000";                      else if(r20000.isSelected()) S += " where price < 20000";                      else if(r25000.isSelected()) S += " where price < 25000";                      else if(r30000.isSelected()) S += " where price < 30000";                      //price 조건을 선택하지 않은 경우 try again label을 보여준다                      else {                          select.setVisible(false);                          success.setVisible(false);                          try\_again.setVisible(true);                      }                  }                  //remove ingredient 기준을 선택했을 때                  else if(remove\_radio.isSelected()) {                      /\*\* 배소연\*/                      //delivery인 경우                      if(flag == 0) S = "select \* from DBCOURSE\_Delivery\_O";                      //No delivery인 경우                      else S = "select \* from DBCOURSE\_Delivery\_X";                      //다중 선택 체크박스 배열                      Checkbox[] check = {meat, seafood, dairy, fungus, vegetable, grain, fruit};                      int count = 0; //선택된 체크박스의 개수와 선택된 ingredient 개수 세는 count 변수                      String S2 = "select name from DBCOURSE\_Ingredient ";                      for(int i=0; i<check.length; i++) {                          if(check[i].getState() == true) {                              count++;                              if(count == 1) S2 += "where type = '" + check[i].getLabel() + "'";                              else S2 += " or type = '" + check[i].getLabel() + "'";                          }                      }                      //아무것도 체크하지 않았을 때                      if(count == 0) S2 += "where type = ''";                      S2 += "group by name";                      count = 0;                      try {                          String temp = "";                          ResultSet rs = stmt.executeQuery(S2);                          while(rs.next()) { //선택된 모든 ingredient를 하나씩 빼는 while문                              count++;                              if(count == 1) {//처음 체크된 경우에만 where문 추가                                  /\*\* sql - 김해리, 신유진\*/                                  if(flag == 0) S += " where food\_name not in " +                                          "(select DBCOURSE\_Delivery\_O.food\_name from DBCOURSE\_Delivery\_O, DBCOURSE\_Ingredient " +                                          "where DBCOURSE\_Delivery\_O.food\_name = DBCOURSE\_Ingredient.food\_name " +                                          "and (DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";                                  else S += S += " where food\_name not in " +                                          "(select DBCOURSE\_Delivery\_X.food\_name from DBCOURSE\_Delivery\_X, DBCOURSE\_Ingredient " +                                          "where DBCOURSE\_Delivery\_X.food\_name = DBCOURSE\_Ingredient.food\_name " +                                          "and (DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";                              }                              else S += " or DBCOURSE\_Ingredient.name = '" + rs.getString(1) + "'";                          }                          if(flag == 0) S += ") group by DBCOURSE\_Delivery\_O.food\_name)";                          else S += ") group by DBCOURSE\_Delivery\_X.food\_name)";                      } catch (SQLException e1) {                          e1.printStackTrace();                      }                      /\*\* 배소연\*/                      //제대로 선택한 경우 success label을 보여준다                      select.setVisible(false);                      try\_again.setVisible(false);                      success.setVisible(true);                  }                  //type 기준을 선택한 경우                  else if(type\_radio.isSelected()) {                      /\*\* sql - 김해리\*/                      //delivery인 경우                      if(flag == 0) S = "select \* from DBCOURSE\_Delivery\_O";                      //No delivery인 경우                      else S = "select \* from DBCOURSE\_Delivery\_X";                      /\*\* 박소연\*/                      //다중 선택 체크박스 배열                      Checkbox[] check = {oriental,western,dessert};                      int count = 0;                      for(int i=0; i<check.length; i++) { //체크한 type을 추가하는 for문                          if(check[i].getState() == true) {                              count++;                              if(count == 1) S += " where type = '" + check[i].getLabel() + "'";                              else S += " or type = '" + check[i].getLabel() + "'";                          }                      }                      /\*\* 배소연\*/                      //아무 것도 체크하지 않았을 때 빈 table을 보여주며 try again label을 보여준다                      if(count == 0) {                          S += " where type = ''";                          select.setVisible(false);                          success.setVisible(false);                          try\_again.setVisible(true);                      }                      //제대로 체크했을 때 success label을 보여준다                      else {                          select.setVisible(false);                          try\_again.setVisible(false);                          success.setVisible(true);                      }                  }                  //recommend 기준을 선택한 경우                  else if(recommend\_radio.isSelected()) {                      /\*\* sql - 김해리\*/                      //delivery인 경우                      if(flag == 0) S = "select restaurant\_name, city, type, food\_name, price " +                              "from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Evaluation";                      //No delivery인 경우                      else S = "select restaurant\_name, city, type, food\_name, price " +                              "from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Evaluation";                      /\*\* 박소연\*/                      //다중 선택 체크박스 배열                      Checkbox[] check = {family,friends,date};                      int count = 0;                      for(int i=0; i<check.length; i++) {//체크한 recommend를 추가하는 for문                          if(check[i].getState() == true) {                              if(count ==0){                                  S += " where recommendation = '" + check[i].getLabel() + "'";                              }                              else S += " or recommendation = '" + check[i].getLabel() + "'";                              count++;                          }                      }                      /\*\* 배소연\*/                      //아무 것도 체크하지 않았을 때 빈 table을 보여주며 try again label을 보여준다                      if(count == 0) {                          S += " where type = ''";                          select.setVisible(false);                          success.setVisible(false);                          try\_again.setVisible(true);                      }                      //제대로 체크했을 때 success label을 보여준다                      else {                          select.setVisible(false);                          try\_again.setVisible(false);                          success.setVisible(true);                      }                      //food\_name으로 group으로 묶는다                      S += " group by food\_name";                  }                  //accompany 기준을 선택한 경우                  else if(accompany\_radio.isSelected()) {                      /\*\* sql - 김해리, 신유진, 박소연\*/                      //delivery인 경우 - only for not delivery의 조건이므로 빈 table과 try again 보여준다                      if(flag == 0) {                          S = "select A.restaurant\_name, A.city, A.type, A.food\_name, A.price " +                                  "from (select \* from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Service)as A, " +                                  "(select \* from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Service) as B " +                                  "where A.restaurant\_name = B.restaurant\_name";                          select.setVisible(false);                          success.setVisible(false);                          try\_again.setVisible(true);                      }                      //No delivery인 경우                      else {                          S = "select A.restaurant\_name, A.city, A.type, A.food\_name, A.price " +                                  "from (select \* from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Service)as A, " +                                  "(select \* from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Service) as B " +                                  "where A.restaurant\_name = B.restaurant\_name";                          int count=0;                          //선택한 조건에 따라 where문 추가                          if(baby.getState()==true){                              S += " and A.permission = 'baby'";                              count++;                          }                          if(pet.getState()==true){                              if(count==0){                                  S += " and B.permission='pet'";                                  count++;                              }                              else S+=" and B.permission = 'pet'";                          }                          if(count==0) S+=" and A.permission = 'none'";                            //food\_name으로 grouping한다                          S += " group by A.food\_name";                            /\*\* 배소연\*/                          //제대로 선택한 경우 success label을 보여준다                          select.setVisible(false);                          try\_again.setVisible(false);                          success.setVisible(true);                      }                  }                  //parking 기준을 선택한 경우                  else if(park\_radio.isSelected()) {                      /\*\* sql - 박소연\*/                      //delivery인 경우 - only for not delivery의 조건이므로 빈 table과 try again 보여준다                      if(flag == 0) {                          S = "select \* from DBCOURSE\_Delivery\_O natural join DBCOURSE\_Address";                          select.setVisible(false);                          success.setVisible(false);                          try\_again.setVisible(true);                      }                      //No delivery인 경우                      else {                          //선택한 조건에 따라 where문을 String S에 추가한다                          S = "select \* from DBCOURSE\_Delivery\_X natural join DBCOURSE\_Address";                          select.setVisible(false);                          try\_again.setVisible(false);                          success.setVisible(true);                          if(park.isSelected()) {                              S += " where parking = 'o'";                          }                          else if(nopark.isSelected()) {                              S += " where parking = 'x'";                          }                          else if(nomatter.isSelected()) {                              S += " where parking = 'o' or parking = 'x'";                          }                          /\*\* 배소연\*/                          //아무것도 선택하지 않았을 때는 빈 table을 보여주며 try again label을 보여준다                          else{                              S += " where parking = 'o' and parking = 'x'";                              select.setVisible(false);                              success.setVisible(false);                              try\_again.setVisible(true);                          }                      }                  }                  /\*\* 배소연\*/                  //아무 기준도 선택하지 않았을 때는 원래의 view를 보여주며 try again label을 보여준다                  else {                      if(flag == 0) S = "select \* from DBCOURSE\_Delivery\_O";                      else S = "select \* from DBCOURSE\_Delivery\_X";                      select.setVisible(false);                      success.setVisible(false);                      try\_again.setVisible(true);                  }                  /\*\* 배소연\*/                  //각 기준에 따라 만들어진 String S로 sql문 실행                  try {                      //table의 모든 tuple을 받아온다                      ResultSet rs = stmt.executeQuery(S);                      //한 tuple씩 받아온다                      while(rs.next()) {                          Object data[] = {rs.getString(1), rs.getString(2), rs.getString(3), rs.getString(4), rs.getInt(5)};                          model.addRow(data);                      }                  } catch (SQLException e1) {                      e1.printStackTrace();                  }                  //user모드에서의 테이블을 보여준다                  showTable\_user();                  //u2 페이지를 보여준다              }              /\*\* 박소연\*/              else if(e.getSource() == nextbutton\_u2) {                  x.setVisible(false);                  noingredient.setVisible(false);                  complete.setVisible(true);                    //테이블이 선택 되었을 때                  if (table.getSelectedRow() > -1){                      //선택된 줄을 받아오고                      int selectRow = table.getSelectedRow();                      String res\_name="";                      String res\_city="";                      String foodtype="";                      String res\_phone="";                      String foodname="";                      String foodprice="";                        //테이블의 정보를 textfield와 choice에 반영되게 한다.                      res\_name=table.getValueAt(selectRow, 0).toString();                      restaurant\_name.setText(res\_name);                      foodtype=table.getValueAt(selectRow, 2).toString();                      food\_type.setText(foodtype);                      foodname=table.getValueAt(selectRow, 3).toString();                      food\_name.setText(foodname);                      foodprice=table.getValueAt(selectRow, 4).toString();                      food\_price.setText(String.valueOf(foodprice)+" won");                      String deliverycost="";                        //delivery인 경우                      if(flag==0){                          try{                              //레스토랑의 delivery cost를 받아온다                              String S = "select delivery\_cost from DBCOURSE\_Delivery where restaurant\_name='"+res\_name+"'";                              ResultSet rs = stmt.executeQuery(S);                              while(rs.next()){                                  deliverycost=String.valueOf(rs.getInt(1));                                  delivery\_cost.setText(deliverycost+" won");                              }                              delivery\_cost\_label.setText("Delivery Cost: ");                          } catch(SQLException e1){                              e1.printStackTrace();                          }                      }                      //No delivery인 경우 delivery cost를 보이지 않도록 설정                      else{                          delivery\_cost.setText("");                          delivery\_cost\_label.setText("");                      }                        /\*\* transaction - 배소연, sql - 배소연, 박소연\*/                      try {                          //setAutoCommit()함수에 false 인자를 주어 transaction을 실행한다                          conn.setAutoCommit(false);                          //음식의 이름이 foodname일때 재료를 구해서 numIngredient가 0인 재료의 개수를 센다                          String S = "select count(ingredient\_name) " +                                  "from (select DBCOURSE\_Ingredient.food\_name, DBCOURSE\_Inventory.ingredient\_name, DBCOURSE\_Inventory.numIngredient " +                                  "from DBCOURSE\_Inventory, DBCOURSE\_Ingredient, DBCOURSE\_Food " +                                  "where (DBCOURSE\_Ingredient.food\_name = DBCOURSE\_Food.name) " +                                  "and (DBCOURSE\_Inventory.ingredient\_name = DBCOURSE\_Ingredient.name) " +                                  "and(DBCOURSE\_Inventory.restaurant\_name = DBCOURSE\_Food.restaurant\_name)) as A " +                                  "where numIngredient = 0 and food\_name = '" + foodname + "'";                          //ingredient가 남아 있는지 확인하기 위한 count(ingredient) tuple 받아오기                          ResultSet rs = stmt.executeQuery(S);                          int count = 0;                          //count 변수를 numIngredient가 0인 재료의 개수로 설정                          if(rs.next()) count = rs.getInt(1);                          if(count == 0) { //재료가 부족하지 않으면                              try {                                  /\*\* 박소연\*/                                  //DBCOURSE\_Order에 insert를 할 때 preparedstatement 이용                                  PreparedStatement pStmt=conn.prepareStatement("insert into DBCOURSE\_Order values(?,?,?,?)");                                  rs = stmt.executeQuery("select max(order\_number) from DBCOURSE\_Order");                                    /\*\* 배소연\*/                                  //주문번호는 max(order\_number)+1로 자동 셋팅                                  if(rs.next()) count = rs.getInt(1);                                  count++;                                    /\*\* 박소연\*/                                  //delivery인 경우                                  if(flag==0){                                      pStmt.setInt(1, count);                                      pStmt.setString(2, res\_name);                                      pStmt.setString(3, foodname);                                      //음식가격에 배달료를 더하여 음식가격을 설정                                      pStmt.setInt(4, (Integer.valueOf(foodprice)+Integer.valueOf(deliverycost)));                                      pStmt.executeUpdate();                                  }                                  //No delivery인 경우                                  else{                                      pStmt.setInt(1, count);                                      pStmt.setString(2, res\_name);                                      pStmt.setString(3, foodname);                                      pStmt.setInt(4, (Integer.valueOf(foodprice)));                                      pStmt.executeUpdate();                                  }                                  /\*\* 배소연\*/                                  //방금 들어간 주문의 재료를 가져온다                                  rs = stmt.executeQuery("select ingredient " +                                          "from DBCOURSE\_Order\_Ingredient " +                                          "where order\_number >= all(select order\_number from DBCOURSE\_Order\_Ingredient)");                                  String ingr = "";                                  while(rs.next()) { //방금 들어간 주문의 레스토랑에서 재료를 하나씩 빼서 요리한다.                                      ingr = rs.getString(1);                                      stmt2.executeUpdate("update DBCOURSE\_Inventory " +                                              "set numIngredient = numIngredient - 1 " +                                              "where DBCOURSE\_Inventory.restaurant\_name = '" + res\_name +                                              "' and DBCOURSE\_Inventory.ingredient\_name = '" + ingr + "' ");                                  }                              } catch (SQLException e1) {                                  e1.printStackTrace();                              }                          }                          else { //재료가 부족하면 X와 noingredient label을 보여준다                              complete.setVisible(false);                              x.setVisible(true);                              noingredient.setVisible(true);                          }                          //수행작업이 잘 이뤄졌을 때 변동사항을 적용한다                          conn.commit();                      } catch (SQLException e1) {                          e1.printStackTrace(); //예외 발생 원인 출력                          if(conn != null) {                              try {                                  //SQLException 발생 시 rollback으로 변동사항을 없는 것으로 설정                                  conn.rollback();                              } catch (SQLException e2) {                                  e2.printStackTrace();                              }                          }                      } finally {                          try { //다시 setAutoCommit을 true로 설정                              conn.setAutoCommit(true);                          } catch (SQLException e1) {                              e1.printStackTrace();                          }                      }                      /\*\* 박소연\*/                      try {                          //레스토랑의 전화번호를 가져온다.                          ResultSet rs = stmt.executeQuery("select phone\_number from DBCOURSE\_restaurant where name="+"'"+res\_name+"'");                          while(rs.next()) {                              res\_phone=rs.getString(1);                          }                      } catch (SQLException e1) {                          e1.printStackTrace();                      }                      //마지막의 order에 보여줄 정보를 설정                      restaurant\_num.setText(res\_phone);                      restaurant\_name.setFont(new Font("Britannic Bold",Font.PLAIN,H/20));                      restaurant\_name.setForeground(Color.white);                      food\_type.setFont(new Font("Britannic Bold",Font.PLAIN,H/20));                      food\_type.setForeground(Color.white);                      food\_price.setFont(new Font("Britannic Bold",Font.PLAIN,H/20));                      food\_price.setForeground(Color.white);                      restaurant\_num.setFont(new Font("Britannic Bold",Font.PLAIN,H/20));                      restaurant\_num.setForeground(Color.white);                      food\_name.setFont(new Font("Britannic Bold",Font.PLAIN,H/20));                      food\_name.setForeground(Color.white);                      delivery\_cost.setFont(new Font("Britannic Bold",Font.PLAIN,H/20));                      delivery\_cost.setForeground(Color.white);                      layout.show(card, "u3");                  }                  /\*\* 배소연\*/                  else{                      //제대로 선택하지 않은 경우 try again label을 보여준다                      select.setVisible(false);                      success.setVisible(false);                      try\_again.setVisible(true);                  }              }              /\*\* 박소연\*/              else if(e.getSource()==getinfobutton){                  if(delete\_radio.isSelected() || update\_radio.isSelected()) {                      //table을 선택했을 때                      if (table.getSelectedRow() > -1){                          //테이블로부터 정보를 받아와서 보여준다.                          int selectRow = table.getSelectedRow();//선택된 줄                          emp\_ID=new String(table.getValueAt(selectRow, 0).toString());                          name.setText(table.getValueAt(selectRow, 1).toString());                          name.setFont(new Font("Courier",Font.PLAIN,H/33));                          c\_res\_name.select(table.getValueAt(selectRow, 2).toString());                          c\_position.select(table.getValueAt(selectRow,3).toString());                          salary.setText(table.getValueAt(selectRow, 4).toString());                          salary.setFont(new Font("Courier",Font.PLAIN,H/33));                          oldname=name.getText();                          oldposition=c\_position.getSelectedItem();                          oldresname=c\_res\_name.getSelectedItem();                          oldsalary=Integer.parseInt(salary.getText());                      }                      else{                          //table을 선택하지 않고 get information button을 눌렀을때 에러메세지를 보여준다                          inform\_emp\_up\_1.setVisible(false);                          inform\_emp\_down\_1.setVisible(false);                          inform\_emp\_up\_done.setVisible(false);                          inform\_emp\_down\_done.setVisible(false);                          inform\_emp\_down\_errtable.setVisible(true);                          inform\_emp\_error.setVisible(true);                          inform\_emp\_down\_format.setVisible(false);                          inform\_emp\_select.setVisible(false);                          inform\_emp\_insert.setVisible(false);                      }                  }                  /\*\* 배소연\*/                  //insert를 선택한 상태인데 getinfo를 누르면 insert label을 보여준다                  else if(insert\_radio.isSelected()){                      inform\_emp\_up\_1.setVisible(false);                      inform\_emp\_down\_1.setVisible(false);                      inform\_emp\_up\_done.setVisible(false);                      inform\_emp\_down\_done.setVisible(false);                      inform\_emp\_down\_errtable.setVisible(false);                      inform\_emp\_error.setVisible(true);                      inform\_emp\_down\_format.setVisible(false);                      inform\_emp\_select.setVisible(false);                      inform\_emp\_insert.setVisible(true);                  }                  //라디오 버튼을 아무것도 선택하지 않고 getinfo를 누르면 select button label을 보여준다                  else {                      inform\_emp\_up\_1.setVisible(false);                      inform\_emp\_down\_1.setVisible(false);                      inform\_emp\_up\_done.setVisible(false);                      inform\_emp\_down\_done.setVisible(false);                      inform\_emp\_down\_errtable.setVisible(false);                      inform\_emp\_error.setVisible(true);                      inform\_emp\_down\_format.setVisible(false);                      inform\_emp\_select.setVisible(true);                      inform\_emp\_insert.setVisible(false);                  }              }              /\*\* 박소연\*/              else if(e.getSource()==enterinfobutton){                  //새로운 정보를 textfield와 choice로부터 받아온다.                  newname=name.getText();                  newposition=c\_position.getSelectedItem();                  newresname=c\_res\_name.getSelectedItem();                  String newsalary\_s=salary.getText();                    //textfield가 비어있을 때 에러메시지 출력                  if(newname.isEmpty()||newsalary\_s.isEmpty()){                      inform\_emp\_up\_1.setVisible(false);                      inform\_emp\_down\_1.setVisible(false);                      inform\_emp\_up\_done.setVisible(false);                      inform\_emp\_down\_done.setVisible(false);                      inform\_emp\_down\_errtable.setVisible(false);                      inform\_emp\_error.setVisible(true);                      inform\_emp\_down\_format.setVisible(true);                      inform\_emp\_select.setVisible(false);                      inform\_emp\_insert.setVisible(false);                  }                  //textfield가 비어있지 않을 때                  else{                      try{                          //salary에 자연수가 아닌 String이 입력되면 exception 처리                          newsalary=Integer.valueOf(newsalary\_s);                            /\*\* 배소연\*/                          model.setRowCount(0);  //테이블 갱신을 위해 초기화                            /\*\* 박소연\*/                          //insert radio button이 눌렸을 때                          if(insert\_radio.isSelected()){                              try {                                  //이전 아이디에 +1 함으로써 새로운 아이디를 자동으로 부여한다                                  int newid=0;                                  ResultSet rs1 = stmt.executeQuery("select max(id) from DBCOURSE\_Employee");                                  if(rs1.next()) newid = Integer.valueOf(rs1.getString(1));                                    //완료메시지를 출력한다                                  inform\_emp\_up\_1.setVisible(false);                                  inform\_emp\_down\_1.setVisible(false);                                  inform\_emp\_up\_done.setVisible(true);                                  inform\_emp\_down\_done.setVisible(true);                                  inform\_emp\_down\_errtable.setVisible(false);                                  inform\_emp\_error.setVisible(false);                                  inform\_emp\_down\_format.setVisible(false);                                  inform\_emp\_select.setVisible(false);                                  inform\_emp\_insert.setVisible(false);                                    //새로운 정보를 table에 insert한다                                  //DBCOURSE\_Employee에 prepared statement를 만든다                                  PreparedStatement pStmt=conn.prepareStatement("insert into DBCOURSE\_Employee values(?,?,?,?,?)");                                  pStmt.setString(1, String.valueOf(newid+1));                                  pStmt.setString(2, newname);                                  pStmt.setString(3, newresname);                                  pStmt.setString(4, newposition);                                  pStmt.setInt(5, newsalary);                                  pStmt.executeUpdate();                              } catch (SQLException e1) {                                  e1.printStackTrace();                              }                          }                          //delete radiobutton을 선택하면                          else if(delete\_radio.isSelected()){                              inform\_emp\_up\_1.setVisible(false);                              inform\_emp\_down\_1.setVisible(false);                              inform\_emp\_up\_done.setVisible(true);                              inform\_emp\_down\_done.setVisible(true);                              inform\_emp\_down\_errtable.setVisible(false);                              inform\_emp\_error.setVisible(false);                              inform\_emp\_down\_format.setVisible(false);                              inform\_emp\_select.setVisible(false);                              inform\_emp\_insert.setVisible(false);                                //employee id로 찾은 employee 정보를 DBCOURSE\_Employee에서 제거한다                              try{                                  stmt.executeUpdate("delete from DBCOURSE\_Employee where id='"+emp\_ID+"'");                              } catch(SQLException se) {                                  se.printStackTrace();                              }                          }                          //update radio button이 선택되었을 때                          else if(update\_radio.isSelected()){                              //안내 메시지를 출력한다.                              inform\_emp\_up\_1.setVisible(false);                              inform\_emp\_down\_1.setVisible(false);                              inform\_emp\_up\_done.setVisible(true);                              inform\_emp\_down\_done.setVisible(true);                              inform\_emp\_down\_errtable.setVisible(false);                              inform\_emp\_error.setVisible(false);                              inform\_emp\_down\_format.setVisible(false);                              inform\_emp\_select.setVisible(false);                              inform\_emp\_insert.setVisible(false);                                /\*\* 배소연 - 트랜잭션, 박소연 - insert\*/                              //delete와 insert가 같이 일어나야 하기 때문에 transaction                              try{                                  //setAutoCommit()함수에 false 인자를 주어 transaction을 실행한다                                  conn.setAutoCommit(false);                                  //dbcourse\_seriesName 테이블에서 사용자가 입력한 시리즈 삭제                                  stmt.executeUpdate("delete from DBCOURSE\_Employee where id='"+emp\_ID+"'");                                  PreparedStatement pStmt=conn.prepareStatement("insert into DBCOURSE\_Employee values(?,?,?,?,?)");                                  pStmt.setString(1, emp\_ID);                                  pStmt.setString(2, newname);                                  pStmt.setString(3, newresname);                                  pStmt.setString(4, newposition);                                  pStmt.setInt(5, newsalary);                                  pStmt.executeUpdate();                                    //수행작업이 잘 이뤄졌을 때 변동사항을 적용한다                                  conn.commit();                              } catch(SQLException se) {                                  se.printStackTrace();                                  if(conn != null) {                                      try {                                          //SQLException 발생 시 rollback으로 변동사항을 없는 것으로 설정                                          conn.rollback();                                      } catch (SQLException e1) {                                          e1.printStackTrace();                                      }                                  }                              } finally {                                  try { //다시 setAutoCommit을 true로 설정                                      conn.setAutoCommit(true);                                  } catch (SQLException e1) {                                      e1.printStackTrace();                                  }                              }                          }                          /\*\*박소연\*/                          else{ //radio button 아무것도 누르지 않고 엔터를 누르면 에러메시지 출력                              inform\_emp\_up\_1.setVisible(false);                              inform\_emp\_down\_1.setVisible(false);                              inform\_emp\_up\_done.setVisible(false);                              inform\_emp\_down\_done.setVisible(false);                              inform\_emp\_down\_errtable.setVisible(false);                              inform\_emp\_error.setVisible(true);                              inform\_emp\_down\_format.setVisible(false);                              inform\_emp\_select.setVisible(true);                              inform\_emp\_insert.setVisible(false);                          }                          /\*\* 배소연\*/                          //수정 이후 결과를 받아오기                          ResultSet rs;                          try {                              rs = stmt.executeQuery("SELECT \* FROM DBCOURSE\_Employee");                              while(rs.next()) {                                  Object data[] = {rs.getString(1), rs.getString(2), rs.getString(3), rs.getString(4), rs.getInt(5)};                                  model.addRow(data);                              }                          } catch (SQLException e1) {                              // TODO Auto-generated catch block                              e1.printStackTrace();                          }                          //바뀐 테이블을 보여준다.                          showTable\_emp();                      } catch(NumberFormatException num){ //Integer를 입력하지 않으면 에러메시지를 보여준다.                          inform\_emp\_up\_1.setVisible(false);                          inform\_emp\_down\_1.setVisible(false);                          inform\_emp\_up\_done.setVisible(false);                          inform\_emp\_down\_done.setVisible(false);                          inform\_emp\_down\_errtable.setVisible(false);                          inform\_emp\_error.setVisible(true);                          inform\_emp\_down\_format.setVisible(true);                          inform\_emp\_select.setVisible(false);                          inform\_emp\_insert.setVisible(false);                      }                  }              }              /\*\* 박소연\*/              //salary를 up하는 경우              else if(e.getSource()==givebutton\_up){                  try{                      String str=plus\_salary.getText();                      Double value=Double.valueOf(str);                        /\*\* 배소연\*/                      model.setRowCount(0); //테이블 갱신을 위해 초기화                        /\*\* 박소연\*/                      if(value<0){ //음수이면 에러메시지 출력                          inform\_emp\_up\_1.setVisible(false);                          inform\_emp\_down\_1.setVisible(false);                          inform\_emp\_up\_done.setVisible(false);                          inform\_emp\_down\_done.setVisible(false);                          inform\_emp\_down\_errtable.setVisible(false);                          inform\_emp\_error.setVisible(true);                          inform\_emp\_down\_format.setVisible(true);                          inform\_emp\_select.setVisible(false);                          inform\_emp\_insert.setVisible(false);                      }                      else{                          //salary 결과를 update한다.                          String S="update DBCOURSE\_Employee set salary=salary\*"+String.valueOf(1+Double.valueOf(str)/(double)100);                          try {                              stmt.executeUpdate(S);                              //수정 이후 결과를 받아오기                              ResultSet rs = stmt.executeQuery("SELECT \* FROM DBCOURSE\_Employee");                              while(rs.next()) {                                  Object data[] = {rs.getString(1), rs.getString(2), rs.getString(3), rs.getString(4), rs.getInt(5)};                                  model.addRow(data);                              }                          } catch (SQLException e1) {                              e1.printStackTrace();                          }                          //완료메세지를 출력한다.                          inform\_emp\_up\_1.setVisible(false);                          inform\_emp\_down\_1.setVisible(false);                          inform\_emp\_up\_done.setVisible(true);                          inform\_emp\_down\_done.setVisible(true);                          inform\_emp\_down\_errtable.setVisible(false);                          inform\_emp\_error.setVisible(false);                          inform\_emp\_down\_format.setVisible(false);                          inform\_emp\_select.setVisible(false);                          inform\_emp\_insert.setVisible(false);                          //table을 보여준다                          showTable\_emp();                      }                  } catch(NumberFormatException num){ //input이 Integer형식이 아니면 에러메세지 출력                      inform\_emp\_up\_1.setVisible(false);                      inform\_emp\_down\_1.setVisible(false);                      inform\_emp\_up\_done.setVisible(false);                      inform\_emp\_down\_done.setVisible(false);                      inform\_emp\_down\_errtable.setVisible(false);                      inform\_emp\_error.setVisible(true);                      inform\_emp\_down\_format.setVisible(true);                      inform\_emp\_select.setVisible(false);                      inform\_emp\_insert.setVisible(false);                  }              }              /\*\* 박소연\*/              //salary를 down하는 경우              else if(e.getSource()==givebutton\_down){                  //textfield로부터 String을 받아온다.                  try{                      String str=minus\_salary.getText();                      Double value=Double.valueOf(str);                        /\*\* 배소연\*/                      model.setRowCount(0);  //테이블 갱신을 위해 초기화                        /\*\* 박소연\*/                      //textfield의 수가 음수이거나 100보다 크면                      if(value>100||value<0){                          //에러메세지 출력                          inform\_emp\_up\_1.setVisible(false);                          inform\_emp\_down\_1.setVisible(false);                          inform\_emp\_up\_done.setVisible(false);                          inform\_emp\_down\_done.setVisible(false);                          inform\_emp\_down\_errtable.setVisible(false);                          inform\_emp\_error.setVisible(true);                          inform\_emp\_down\_format.setVisible(true);                          inform\_emp\_select.setVisible(false);                          inform\_emp\_insert.setVisible(false);                      }                      else{ //0과 100 사이의 수라면 salary update                          String S="update DBCOURSE\_Employee set salary=salary\*"+String.valueOf(1-Double.valueOf(str)/(double)100);                          try {                              stmt.executeUpdate(S);                              //수정 이후의 결과를 받아온다                              ResultSet rs = stmt.executeQuery("SELECT \* FROM DBCOURSE\_Employee");                              while(rs.next()) {                                  Object data[] = {rs.getString(1), rs.getString(2), rs.getString(3), rs.getString(4), rs.getInt(5)};                                  model.addRow(data);                              }                          } catch (SQLException e1) {                              e1.printStackTrace();                          }                          //완료메세지 출력                          inform\_emp\_up\_1.setVisible(false);                          inform\_emp\_down\_1.setVisible(false);                          inform\_emp\_up\_done.setVisible(true);                          inform\_emp\_down\_done.setVisible(true);                          inform\_emp\_down\_errtable.setVisible(false);                          inform\_emp\_error.setVisible(false);                          inform\_emp\_down\_format.setVisible(false);                          inform\_emp\_select.setVisible(false);                          inform\_emp\_insert.setVisible(false);                          //새로운 테이블을 보여준다.                          showTable\_emp();                      }                  }catch(NumberFormatException num){ //input이 Integer형식이 아니면 exception 처리                      inform\_emp\_up\_1.setVisible(false);                      inform\_emp\_down\_1.setVisible(false);                      inform\_emp\_up\_done.setVisible(false);                      inform\_emp\_down\_done.setVisible(false);                      inform\_emp\_down\_errtable.setVisible(false);                      inform\_emp\_error.setVisible(true);                      inform\_emp\_down\_format.setVisible(true);                      inform\_emp\_select.setVisible(false);                      inform\_emp\_insert.setVisible(false);                  }              }              /\*\* 박소연\*/              //salary를 기준으로 직원을 해고하는 경우              else if(e.getSource()==firebutton){                  //textfield로부터 string을 받아온다                  try{                      String str=delete\_salary.getText();                      Double value=Double.valueOf(str);                        /\*\* 배소연\*/                      model.setRowCount(0);  //테이블 갱신을 위해 초기화                        /\*\* 박소연\*/                      String S="delete from DBCOURSE\_Employee where salary>"+String.valueOf(value);                        //textfield에 입력된 수보다 큰 salary를 가진 employee는 delete한다.                      try {                          stmt.executeUpdate(S);                          //수정 이후의 결과를 받아온다.                          ResultSet rs = stmt.executeQuery("SELECT \* FROM DBCOURSE\_Employee");                          while(rs.next()) {                              Object data[] = {rs.getString(1), rs.getString(2), rs.getString(3), rs.getString(4), rs.getInt(5)};                              model.addRow(data);                          }                      } catch (SQLException e1) {                          e1.printStackTrace();                      }                      //완료 메세지를 출력한다.                      inform\_emp\_up\_1.setVisible(false);                      inform\_emp\_down\_1.setVisible(false);                      inform\_emp\_up\_done.setVisible(true);                      inform\_emp\_down\_done.setVisible(true);                      inform\_emp\_down\_errtable.setVisible(false);                      inform\_emp\_error.setVisible(false);                      inform\_emp\_down\_format.setVisible(false);                      inform\_emp\_select.setVisible(false);                      inform\_emp\_insert.setVisible(false);                      //새로운 테이블을 보여준다.                      showTable\_emp();                  } catch(NumberFormatException num){ //input형식이 Integer가 아니면 exception 처리한다                      inform\_emp\_up\_1.setVisible(false);                      inform\_emp\_down\_1.setVisible(false);                      inform\_emp\_up\_done.setVisible(false);                      inform\_emp\_down\_done.setVisible(false);                      inform\_emp\_down\_errtable.setVisible(false);                      inform\_emp\_error.setVisible(true);                      inform\_emp\_down\_format.setVisible(true);                      inform\_emp\_select.setVisible(false);                      inform\_emp\_insert.setVisible(false);                  }              }              /\*\* 배소연\*/              //선택한 order를 처리하여 profit을 올리는 버튼              else if (e.getSource() == takeorderbutton) {                  //order table과 profit을 보여주는 테이블을 초기화한다                  model.setRowCount(0);                  model2.setRowCount(0);                  //this is order list label을 보여준다                  empty.setVisible(false);                  profit.setVisible(false);                  thisis.setVisible(true);                  order.setVisible(true);                  String res = ""; String ingr = ""; int price = 0;                  //order 처리와 profit update는 항상 같이 이뤄져야 하기 때문에 transaction                  try {                      //setAutoCommit()함수에 false 인자를 주어 transaction을 실행한다                      conn.setAutoCommit(false);                      //order number가 가장 작은 tuple을 고른다                      ResultSet rs = stmt.executeQuery("select restaurant\_name, price from DBCOURSE\_Order " +                              "where order\_number <= all(select order\_number from DBCOURSE\_Order)");                      if(rs.next()) {                          res = rs.getString(1);                          price = rs.getInt(2);                      }                      //가져온 restaurant name과 price로 profit을 update한다                      stmt.executeUpdate("update DBCOURSE\_Restaurant " +                              "set profit = profit + " + price +                              " where name = '" + res + "'");                      //처리가 끝난 order를 order table에서 delete한다                      stmt2.executeUpdate("delete from DBCOURSE\_Order " +                              "where order\_number <= all(select order\_number from (select order\_number from DBCOURSE\_Order) as A)");                        //남은 order table의 tuple을 가져온다                      rs = stmt.executeQuery("select \* from DBCOURSE\_Order");                      while(rs.next()) {                          Object data[] = {rs.getInt(1), rs.getString(2), rs.getString(3), rs.getInt(4)};                          model.addRow(data);                      }                      //update된 restaurant 별 현재의 profit을 가져온다                      rs = stmt.executeQuery("select name as restaurant\_name, profit from DBCOURSE\_Restaurant");                      while(rs.next()) {                          Object data[] = {rs.getString(1), rs.getInt(2)};                          model2.addRow(data);                      }                      //수행작업이 잘 이뤄졌을 때 변동사항을 적용한다                      conn.commit();                  } catch (SQLException e1) {                      e1.printStackTrace();                      if(conn != null) {                          try {                              //SQLException 발생 시 rollback으로 변동사항을 없는 것으로 설정                              conn.rollback();                          } catch (SQLException e2) {                              e2.printStackTrace();                          }                      }                  } finally {                      try { //다시 setAutoCommit을 true로 설정                          conn.setAutoCommit(true);                      } catch (SQLException e1) {                          e1.printStackTrace();                      }                  }                  //order table과 restaurant 별 profit table을 보여준다                  showTable\_order();                  showTable\_profit();              }              /\*\* 배소연\*/              //영업 종료를 하고 직원들에게 daily salary를 지급하고 최종 profit을 계산하는 버튼              else if(e.getSource() == closebutton) {                  try {                      int count = 0;                      //order table이 다 처리 되어 비어 있는 상태인지 확인                      ResultSet rs = stmt.executeQuery("select count(order\_number) from DBCOURSE\_Order");                      if(rs.next()) count = rs.getInt(1);                      //order table에 아직 주문이 남아있으면 list를 비우라는 label을 보여준다                      if(count != 0) {                          thisis.setVisible(false);                          order.setVisible(false);                          profit.setVisible(false);                          empty.setVisible(true);                      }                      //order table이 비어있으면                      else {                          model2.setRowCount(0);                          thisis.setVisible(true);                          order.setVisible(false);                          empty.setVisible(false);                          profit.setVisible(true);                            //레스토랑 별 sum(salary)를 구한다                          rs = stmt.executeQuery("select restaurant\_name, sum(salary) " +                                  "from DBCOURSE\_Employee " +                                  "group by restaurant\_name");                          String res = ""; int sum = 0;                          while(rs.next()) { //가져온 restaurant name과 sum(salary)로 profit을 update한다                              res = rs.getString(1); sum = rs.getInt(2);                              stmt2.executeUpdate("update DBCOURSE\_Restaurant " +                                      "set profit = profit - " + sum +                                      " where name = '" + res + "'");                          }                          //update된 table을 보여준다                          rs = stmt.executeQuery("select name as restaurant\_name, profit from DBCOURSE\_Restaurant");                          while(rs.next()) {                              Object data[] = {rs.getString(1), rs.getInt(2)};                              model2.addRow(data);                          }                          showTable\_profit();                      }                  } catch (SQLException e1) {                      e1.printStackTrace();                  }              }          }          /\*\* 배소연\*/          //user모드의 table을 보여주는 메소드          void showTable\_user() {              //defaultTableModel을 기반으로 JTable 생성              table = new JTable(model);              table.getTableHeader().setBackground(Color.WHITE);              table.getTableHeader().setFont(new Font("Britannic Bold",Font.PLAIN,H/35));              table.setSelectionBackground(Color.decode("#FCE4EC"));              table.setFont(new Font("Courier",Font.PLAIN,H/35));              table.getColumn("restaurant").setPreferredWidth(W/15);              table.getColumn("city").setPreferredWidth(W/15);              table.getColumn("type").setPreferredWidth(W/40);              table.getColumn("food").setPreferredWidth(W/10);              table.getColumn("price").setPreferredWidth(W/50);              //price column 오른쪽 정렬              DefaultTableCellRenderer dtcr = new DefaultTableCellRenderer();              dtcr.setHorizontalAlignment(SwingConstants.RIGHT);              table.getColumn("price").setCellRenderer(dtcr);              table.setRowHeight(H/30);              //table 헤더를 눌렀을 경우 정렬              table.setAutoCreateRowSorter(true);              //JTable을 담은 JScrollPane 생성              jsp = new JScrollPane(table);              jsp.setPreferredSize(new Dimension(W/2,H/2));              //tablepanel의 모든 컴포넌트 삭제              tablepanel\_u.removeAll();              //tablepanel의 위치 설정              tablepanel\_u.setBounds(7\*W/15, H/10, W/2, H/2);              //tablepanel에 JTable을 담은 JScrollPane 추가              tablepanel\_u.add(jsp);              //tablepanel의 UI reset              tablepanel\_u.updateUI();              //tablepanel의 배경을 투명하게 설정              tablepanel\_u.setOpaque(false);          }          //manager모드의 employee management 페이지에서의 table을 보여주는 메소드          void showTable\_emp() {              //defaultTableModel을 기반으로 JTable 생성              table = new JTable(model);              table.getTableHeader().setBackground(Color.WHITE);              table.getTableHeader().setFont(new Font("Britannic Bold",Font.PLAIN,H/35));              table.setSelectionBackground(Color.decode("#FCE4EC"));              table.setFont(new Font("Courier",Font.PLAIN,H/35));              table.getColumn("ID").setPreferredWidth(W/60);              table.getColumn("name").setPreferredWidth(W/10);              table.getColumn("restaurant name").setPreferredWidth(W/15);              table.getColumn("position").setPreferredWidth(W/40);              table.getColumn("salary(daily)").setPreferredWidth(W/20);              //salary column 오른쪽 정렬              DefaultTableCellRenderer dtcr = new DefaultTableCellRenderer();              dtcr.setHorizontalAlignment(SwingConstants.RIGHT);              table.getColumn("salary(daily)").setCellRenderer(dtcr);              table.setRowHeight(H/30);              //table 헤더를 눌렀을 경우 정렬              table.setAutoCreateRowSorter(true);              //JTable을 담은 JScrollPane 생성              jsp = new JScrollPane(table);              jsp.setPreferredSize(new Dimension(2\*W/3,3\*H/5));              //tablepanel의 모든 컴포넌트 삭제              tablepanel\_emp.removeAll();              //tablepanel의 위치 설정              tablepanel\_emp.setBounds(4\*W/15, H/25, 2\*W/3, 3\*H/5);              //tablepanel에 JTable을 담은 JScrollPane 추가              tablepanel\_emp.add(jsp);              //tablepanel의 UI reset              tablepanel\_emp.updateUI();              //tablepanel의 배경을 투명하게 설정              tablepanel\_emp.setOpaque(false);          }          //manager모드의 order processing 페이지에서의 order table을 보여주는 메소드          void showTable\_order() {              //defaultTableModel을 기반으로 JTable 생성              table = new JTable(model);              table.getTableHeader().setBackground(Color.WHITE);              table.getTableHeader().setFont(new Font("Britannic Bold",Font.PLAIN,H/35));              table.setSelectionBackground(Color.decode("#FCE4EC"));              table.setFont(new Font("Courier",Font.PLAIN,H/35));              table.getColumn("number").setPreferredWidth(W/50);              table.getColumn("restaurant\_name").setPreferredWidth(W/15);              table.getColumn("food\_name").setPreferredWidth(W/10);              table.getColumn("price").setPreferredWidth(W/50);              //price column 오른쪽 정렬              DefaultTableCellRenderer dtcr = new DefaultTableCellRenderer();              dtcr.setHorizontalAlignment(SwingConstants.RIGHT);              table.getColumn("price").setCellRenderer(dtcr);              table.setRowHeight(H/30);              //table 헤더를 눌렀을 경우 정렬              table.setAutoCreateRowSorter(true);              //JTable을 담은 JScrollPane 생성              jsp = new JScrollPane(table);              jsp.setPreferredSize(new Dimension(11\*W/20,H/3));              //tablepanel의 모든 컴포넌트 삭제              tablepanel\_order.removeAll();              //tablepanel의 위치 설정              tablepanel\_order.setBounds(W/30, H/10, 11\*W/20, H/3);              //tablepanel에 JTable을 담은 JScrollPane 추가              tablepanel\_order.add(jsp);              //tablepanel의 UI reset              tablepanel\_order.updateUI();              //tablepanel의 배경을 투명하게 설정              tablepanel\_order.setOpaque(false);          }          //manager모드의 order processing 페이지에서의 profit table을 보여주는 메소드          void showTable\_profit() {              //defaultTableModel을 기반으로 JTable 생성              table2 = new JTable(model2);              table2.getTableHeader().setBackground(Color.WHITE);              table2.getTableHeader().setFont(new Font("Britannic Bold",Font.PLAIN,H/35));              table2.setSelectionBackground(Color.decode("#FCE4EC"));              table2.setFont(new Font("Courier",Font.PLAIN,H/35));              table2.getColumn("restaurant\_name").setPreferredWidth(W/15);              table2.getColumn("profit").setPreferredWidth(W/50);              //profit column 오른쪽 정렬              DefaultTableCellRenderer dtcr = new DefaultTableCellRenderer();              dtcr.setHorizontalAlignment(SwingConstants.RIGHT);              table2.getColumn("profit").setCellRenderer(dtcr);              table2.setRowHeight(H/30);              //table 헤더를 눌렀을 경우 정렬              table2.setAutoCreateRowSorter(true);              //JTable을 담은 JScrollPane 생성              jsp2 = new JScrollPane(table2);              jsp2.setPreferredSize(new Dimension(W/3, 2\*H/5));              //tablepanel의 모든 컴포넌트 삭제              tablepanel\_profit.removeAll();              //tablepanel의 위치 설정              tablepanel\_profit.setBounds(31\*W/50, H/10, W/3, 2\*H/5);              //tablepanel에 JTable을 담은 JScrollPane 추가              tablepanel\_profit.add(jsp2);              //tablepanel의 UI reset              tablepanel\_profit.updateUI();              //tablepanel의 배경을 투명하게 설정              tablepanel\_profit.setOpaque(false);          }            @Override          public void itemStateChanged(ItemEvent arg0) {}      }  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |