Графический интерфейс пользователя. Обзор визуальных компонентов Swing.

Контейнеры высшего уровня

1. JWindow - окно без рамки
2. JFrame - окно с рамкой
3. JDialog - диалоговое окно
4. JApplet - апплет

Окно без рамки JWindow

Методы	Описание
setLocation(), setSize(), setBounds()	Эта группа методов позволяет задать позицию и размеры окна на экране. Первый метод задает позицию окна, второй позволяет указать его размеры, а с помощью третьего можно сразу задать прямоугольник, который займет окно на экране
pack()	Позволяет «упаковать» имеющиеся в окне компоненты, так чтобы они занимали столько места, сколько им необходимо, а не больше и не меньше.
show(), setVisible()	Выводят окно на экран. После вызова этих методов компоненты переходят в «видимое» состояние и начинают обслуживаться очередью событий. Метод show() к тому же проводит валидацию содержащихся в окне компонентов.
dispose()	Убирает окно с экрана (если оно в момент вызова метода видимо) и освобождает все принадлежащие ему ресурсы. Если в приложении много окон, во избежание нехватки памяти следует вызывать этот метод для тех из них, что больше не используются

Окно с рамкой JFrame

1. Является наследником JWindow
2. Упрощенное закрытие окна setDefaultCloseOperation()
3. setUndecorated() - возможность убрать рамку и элементы управления
4. Доступные слушатели событий:
    1. WindowListener
    2. WindowFocusListener

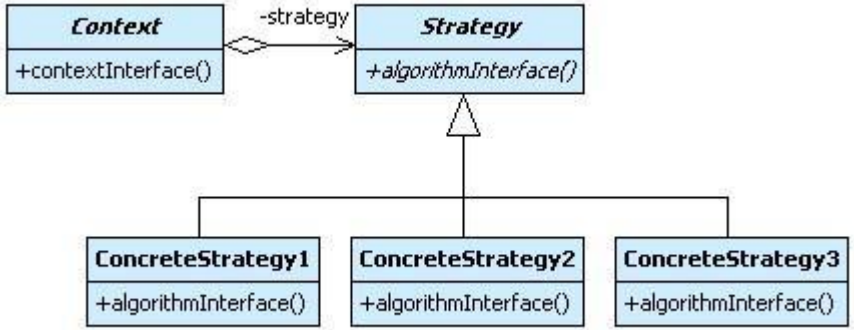
Диалоговое окно JDialog

1. Может быть модальным и немодальным
2. Позволяет указать родительское окно
3. Также имеет метод setDefaultCloseOperation() для упрощения закрытия
4. Для удобства имеет перегруженный конструктор

Менеджеры компоновки

1. BorderLayout
2. FlowLayout
3. GridLayout
4. GridBagLayout
5. SpringLayout
6. BoxLayout

Паттерн “Стратегия”



- Программа должна обеспечивать различные варианты алгоритма
- Нужно изменять поведение каждого экземпляра класса
- Необходимо изменять поведение объектов на стадии выполнения
- Введение интерфейса позволяет классам-клиентам ничего не знать о классах, реализующих этот интерфейс и инкапсулирующих в себе конкретные алгоритмы

java.awt.LayoutManager

Учитывает следующие параметры компонентов

- Предпочтительный размер
- Минимальный размер
- Максимальный размер
- Выравнивание по осям X и Y
- Границы контейнера

Полярное расположение BorderLayout

1. Является менеджером по умолчанию для JWindow, JFrame, JDialog
2. Использование: add(component, область)
3. Виды областей:
    1. BorderLayout.NORTH
    2. BorderLayout.SOUTH
    3. BorderLayout.WEST
    4. BorderLayout.EAST
    5. BorderLayout.CENTER

Последовательное расположение FlowLayout

1. Создает последовательное размещение объектов с указанным выравниванием и заданными между этими объектами интервалами.
2. В случае нехватки места переходит на следующую строку
3. Позволяет задавать расстояния между элементами по вертикали и горизонтали.
4. Позволяет указать горизонтальное выравнивание компонентов

Табличное расположение GridLayout

1. Разделяет контейнер на таблицу с ячейками одинакового размера.
2. Все компоненты всегда выводятся на экран, как бы ни было велико или мало доступное пространство.
3. Объекты располагаются последовательно слева направо.

java.awt.GridBagLayout

Создает гибкое табличное размещение объектов, позволяя размещать один компонент в нескольких ячейках.
GridBagConstraints - определят параметры размещения отдельных компонент:

- gridx и gridy - номер столбца и номер строки для ячейки
- gridwidth и gridheight - количество ячеек, занимаемых добавляемым компонентом.
- fill - стратегия распределения компоненту свободного пространства (GridBagConstraints.NONE, GridBagConstraints.BOTH, GridBagConstraints.HORIZONTAL, GridBagConstraints.VERTICAL)

GridBagConstraints

- anchor - выравнивание компонента внутри отведенного для него пространства (GridBagConstraints.CENTER, .., GridBagConstraints.SOUTHWEST)
- weightx и weighty - стратегию изменения размеров компонента (значения от 0.0 до 1.0)
- ipadx и ipady - позволяют указать, что размеры компонента необходимо увеличить на заданное количество пикселов
- insets - позволяет задать для компонента отступы от краев выделенной ему области (класс Insets)

SpringLayout

1. С каждым компонентом ассоциируется особый информационный объект Spring, который позволяет задать расстояние (в пикселах) между парой границ различных компонентов.
2. Границ у компонента четыре — это его северная, восточная, западная и южная стороны.
3. Позволяет задавать расстояние и между границами одного и того же компонента
4. По умолчанию все компоненты имеют предпочтительный размер
5. SpringLayout тщательно учитывает и два остальных размера, не делая компоненты меньше минимального и больше максимального размеров.
6. предназначен прежде всего для автоматизированного построения интерфейса

Блочное расположение BoxLayout

1. выкладывает компоненты в контейнер блоками: столбиком (по оси Y) или полоской (по оси X)
2. каждый отдельный компонент можно выравнивать по центру, по левому или по правому краям, а так же по верху или по низу
3. Расстояние между компонентами по умолчанию нулевое, но для его задания существуют это распорки (struts), заполнители (glues) и фиксированные области (rigid areas).
4. Имеется возможность создавать через методы класса Box
    1. public static Box createHorizontalBox()
    2. public static Box createVerticalBox()

Надписи JLabel

Свойства	Предназначение
verticalAlignment, horizontalAlignment (и соответствующие методы get/set)	Позволяют задать расположение содержимого надписи (и текста, и значка) относительно ее границ. Возможные положения описаны константами из интерфейса SwingContants, у вас есть три варианта расположения по вертикали (центр, верх, низ) и столько же по горизонтали (центр, справа, слева). Всего получается девять возможных позиций
verticalTextPosition, horizontalTextPosition (и соответствующие методы get/set)	Дают возможность указать надписи, как ей следует располагать текст относительно значка. Вы можете задать положение текста по вертикали (по центру значка, по его верхней или нижней границе) и по горизонтали (по центру значка, слева или справа от него). В итоге также получается девять вариантов (можно даже разместить текст поверх значка)
iconTextGap	В нашем примере мы не использовали это свойство, но оно очень просто и позволяет указать расстояние между текстом и значком в пикселах

Рамки

Название рамки	Назначение
TitledBorder	Позволяет совместить любую рамку и текст, описывающий то, что в этой рамке находится. При этом текст может находиться на любой стороне рамки и иметь различные варианты выравнивания. Очень часто используется в интерфейсах
LineBorder	Одна из самых простых, но наиболее часто используемых рамок. Рисует рамку линией определенного цвета, толщину линии можно выбрать, позволяет скруглять углы
EmptyBorder	позволяет окружить компонент пустыми полями со всех четырех сторон.
EtchedBorder	Рамка с тиснением (в стиле чеканки по металлу). Может быть вогнутой или выпуклой

Рамки (продолжение)

Название рамки	Назначение
BevelBorder	Позволяет создать объемную рамку, выпуклую или вогнутую, по желанию можно настроить цвета, требуемые для получения объемных эффектов.
SoftBevelBorder	Эта рамка аналогична предыдущей (унаследована от нее), обладает теми же свойствами, но дополнительно позволяет скруглить углы рамки
CompoundBorder	Позволяет совместить две рамки и избавляет от утомительного совмещения панелей. Вкладывая такие рамки друг в друга, можно совместить их произвольное количество
MatteBorder	позволяет использовать узор из значков Icon. С ее помощью можно создавать очень необычные рамки. Она часто избавляет от необходимости создавать свои процедуры прорисовки и новые рамки

Фабрика BorderFactory

```
static Border createBevelBorder(int type)
static CompoundBorder createCompoundBorder()
static Border createEmptyBorder()
static Border createEtchedBorder()
static Border createLineBorder(Color color)
static TitledBorder createTitledBorder(String title)
static Border createSoftBevelBorder(int type)
static MatteBorder createMatteBorder(int top, int left, int bottom, int right, Color
color)
```

Элементы управления

Кнопки JButton

Свойства (и методы get/set)	Описание
icon	Используется для установки «повседневного» значка, который будет виден всегда, когда кнопка доступна
rolloverIcon	С помощью этого свойства можно установить значок для получения эффекта «наведения мыши» — когда указатель мыши оказывается на кнопке, появляется этот значок.
pressedIcon	Используется для установки значка, который выводится на экран при щелчке пользователем на кнопке
disabledIcon	Если задать значок посредством данного свойства, то, когда кнопка будет отключена с помощью метода setEnabled(false), появится специальный значок. Если специального значка нет, а обычный значок есть, то в качестве значка для отключенной кнопки будет использована черно-белая копия обычного значка

JButton

Свойства (и методы get/set)	Описание
margin	Это свойство позволяет управлять размером полей, которые отделяют содержимое кнопки от ее границ.
verticalAlignment, horizontalAlignment	Данная пара свойств используется для изменения позиции всего содержимого кнопки (и значка, и текста) относительно границ кнопки. С их помощью поместить это содержимое можно в любой из углов кнопки или в ее центр (по умолчанию оно там и находится).
horizontalTextPosition, verticaiTextPosition	С помощью этих свойств можно управлять положением текста кнопки относительно ее значка по горизонтали и вертикали.
iconTextGap	Свойство позволяет изменить расстояние между значком и текстом

Элементы управления с двумя состояниями

1. Выключатели JToggleButton
2. Группы элементов управления ButtonGroup
3. Переключатели JRadioButton
4. Флажки JCheckBox

Обычные списки JList

1. позволяют выводить на экран перечень некоторых элементов
2. использует две модели:
    1. для формирования списка показываемых элементов (ListModel -> DefaultListModel)
    2. для показа выделенных (ListSelectionModel -> DefaultListSelectionModel)
3. Позволяет настраивать внешний вид в широких пределах:
    1. установкой свойств
    2. ListCellRenderer (метод getListCellRendererComponent())

Раскрывающиеся списки JComboBox

1. служат для выбора одного из множества доступных вариантов
2. виден только сам выбранный элемент
3. Список возможных альтернатив выводится в специальном всплывающем меню, которое после выбора скрывается
4. допускают редактирование текущего элемента
5. обладает встроенной возможностью поиска элементов с клавиатуры
6. требует только одну модель ComboBoxModel

Диапазоны значений

1. Ползунки JSlider
2. Индикаторы процесса JProgressBar
3. Счетчики JSpinner

Ползунки JSlider

1. позволяет плавно изменять выбираемое в некотором диапазоне значение с помощью специального регулятора,
2. требуется специальная модель BoundedRangeModel, хранящая информацию об ограниченном наборе данных. Данные в ней хранятся в числовом виде в четырех целых (int) числах.
    1. минимальное значение (свойство с названием minimum),
    2. максимальное значение (maximum),
    3. текущее значение (value)
    4. внутренний диапазон (extent)

Индикаторы процесса JProgressBar

1. демонстрируют пользователю, на каком этапе выполнения находится некий процесс.
2. использует BoundedRangeModel, предназначенную для хранения ограниченного диапазона чисел.

Свойства JProgressBar

Свойства	Описание
orientation	Управляет ориентацией индикатора. Ориентация может быть вертикальной или горизонтальной
stringPainted	Позволяет включить или выключить прорисовку текста на индикаторе процесса. По умолчанию прорисовка текста выключена. Если вы ее включаете, то стандартно будет прорисовываться значение в процентах, показывающее, какая часть процесса закончена (значения берутся из модели). Сменить текст позволяет следующее свойство
string	Задает текст, который будет прорисовываться на индикаторе процесса.
value, maximum, minimum	Эти свойства позволяют манипулировать данными модели BoundedRangeModel напрямую с помощью самого индикатора

Счетчики JSpinner

1. позволяют пользователю сделать выбор из набора альтернатив, которые он в поиске нужного значения может «прокручивать» в любую сторону.
2. Использует модель SpinnerModel, которая позволяет узнать текущее значение счетчика, следующее и предыдущее значения, сменить текущее значение новым, а также поддерживает списки слушателей ChangeListener, которые оповещаются, когда меняется текущее значение модели.
3. Позволяет организовать выбор дат с использованием SpinnerDateModel.

Управление пространством

1. Панель с вкладками JTabbedPane
2. Разделяемая панель JSplitPane
3. Панель прокрутки JScrollPane

Панель с вкладками JTabbedPane

1. позволяет выводить на экран вкладки (tabs)
2. вкладка добавляется при вызове метода add() или addTab()

tabPlacement	Задает расположение ярлычков вкладок у одной из четырех сторон панели. По умолчанию (если вызывается конструктор JTabbedPane без параметров) ярлычки располагаются вверху панели (TOP)
tabLayoutPolicy	Определяет, как будут располагаться вкладки в том случае, если в контейнере не хватит места для их размещения в один ряд.
titleAt, iconAt	позволяют задать надпись и значок для ярлычка любой вкладки
toolTipTextAt	всплывающая подсказка
foregroundAt, backgroundAt	цвет шрифта и фона отдельной вкладки

Разделяемая панель JSplitPane

Свойства	Описание
orientation	Задает ориентацию разделяемой панели. Разделяемая панель может быть горизонтальной или вертикальной
continuousLayout	Позволяет включить или выключить режим непрерывного обновления компонентов при перемещении вешки разбивки
dividerSize	Управляет размером (в пикселах) вешки разбивки
oneTouchExpandable	добавление кнопок скрытия панелей
leftComponent, topComponent	установка левого или верхнего компонента
rightComponent, bottomComponent	установка правого или нижнего компонента

Панель прокрутки JScrollPane

1. позволяет наделить компонент (обычно большого размера) возможностью прокрутки
2. Позволяет задавать политику появления скроллов:
    1. по вертикали
        1. `VERTICAL_SCROLLBAR_AS_NEEDED`
        2. `VERTICAL_SCROLLBAR_NEVER`
        3. `VERTICAL_SCROLLBAR_ALWAYS`
    2. по горизонтали
        1. `HORIZONTAL_SCROLLBAR_AS_NEEDED`
        2. `HORIZONTAL_SCROLLBAR_NEVER`
        3. `HORIZONTAL_SCROLLBAR_ALWAYS`

Текстовые компоненты

1. Текстовые поля
2. Многострочное поле JTextArea
3. Компонент JFormattedTextField

Текстовые поля

Имеет два вида - JTextField и JPasswordField

Свойства	Описание
text	Позволяет получить введенный в поле текст или заменить его. Для поля с конфиденциальной информацией лучше использовать метод getPassword()
columns	Задает количество столбцов в поле
font	Определяет используемый в текстовом поле шрифт.
horizontalAlignment	Управляет выравниванием текста в поле
echoChar (JPasswordField)	Задает символ-заменитель для ввода секретной информации. По умолчанию используется символ звездочки (*)

Многострочное поле JTextArea

Свойства	Описание
rows, columns	Задают размеры многострочного текстового поля, в строках и символах соответственно
lineWrap, wrapStyleWord	Управляют включением переноса текста по строкам и типом этого переноса
font	Позволяет задать шрифт для многострочного текстового поля
tabSize	Контролирует размер символа табуляции в текстовом поле. По умолчанию для символа табуляции используются 8 «обычных» символов
lineCount, iineOfOffset,, lineStartOffset, lineEndOffset (readonly)	Данные методы позволяют получить исчерпывающую информацию о распределении текста многострочного поля по строкам

Форматированный вывод — компонент JFormattedTextField

1. позволяет выводить данные согласно специальным текстовым форматам и маскам
2. ограничивает ввод пользователя, разрешая тому вводить данные только в соответствии с заданным в поле форматом.
3. Имеет фоматирующий объект, наследующий абстрактный внутренний класс AbstractFonnatter.
4. Стандартные форматирующие объекты
    1. MaskFormatter
    2. DateFormatter
    3. NumberFormatter

JTable

1. Дает возможность с легкостью выводить двухмерную информацию
2. Позволяет настраивать и сортировать данные
3. Позволяет управлять заголовками таблицы
4. Имеет ряд вспомогательных классов из пакета javax.swing.table
5. Оперирует данными с помощью модели tableModel

TableModel

Название метода	Описание
getRowCount()	количество строк в таблице
getColumnCount()	количество столбцов в таблице
getValueAt(row, col)	Позволяет указать данные в ячейке. Данные могут иметь любой тип. Отсчет строк и столбцов ведется с нуля
getColumnName(col)	позволяет задать отображаемое имя для столбца
isCellEditable(row, col)	позволяет указать, можно ли редактировать ячейку в таблице с указанным местоположением.
setValueAt(val, row, col)	Метод используется для изменения значения ячейки таблицы
getColumnClass(col)	Позволяет задать тип данных, хранимых в столбце (тип задается в виде объекта Class)

Литература

1. K.Arulkumaran, A.Sivayini. Java/J2EE Job Interview Companion
2. Портянкин И. Swing. Эффективные пользовательские интерфейсы
3. Java™ Platform, Standard Edition 7 API Specification

Домашнее задание

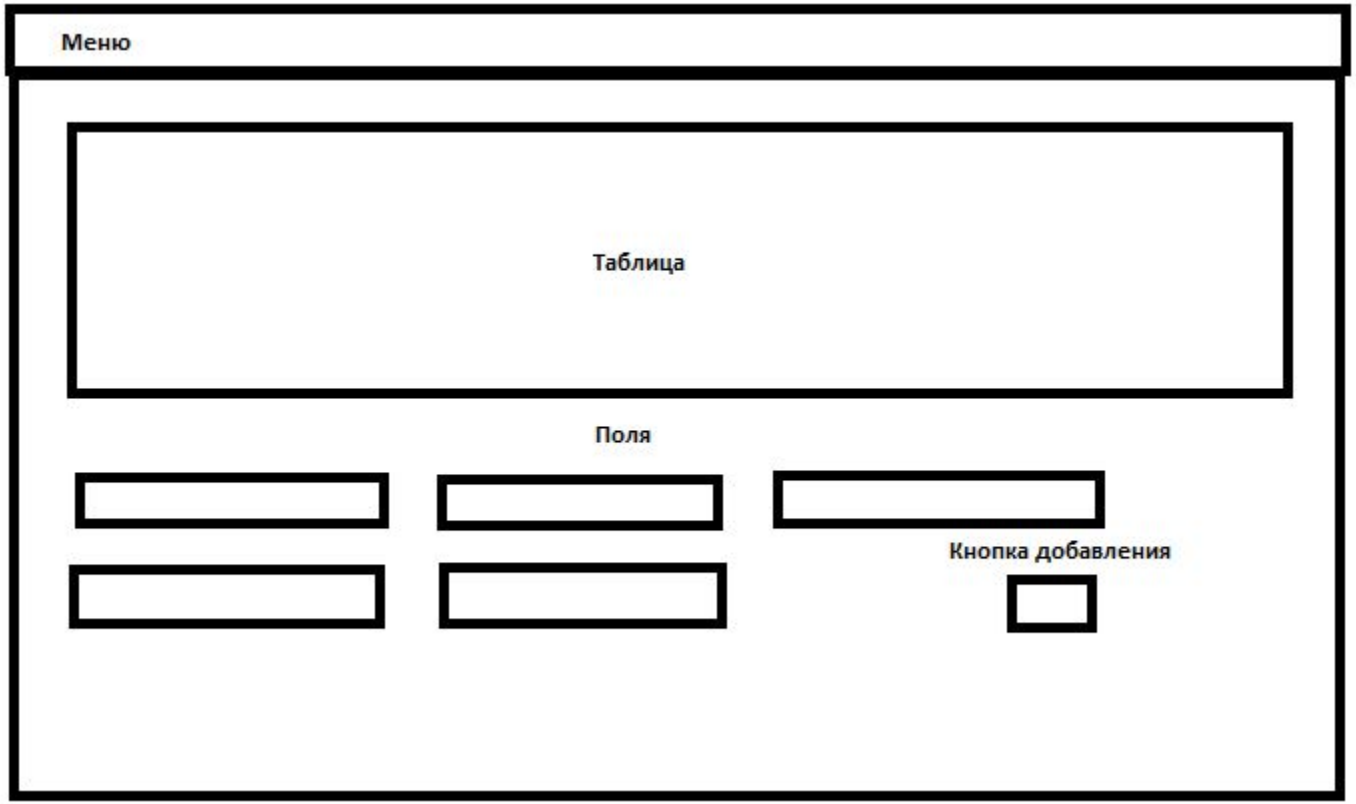
Разработать приложение, которое выводит на экран таблицу с книгами. Количество столбцов таблицы должно быть не менее 5, например:

1. Название;
2. Автор;
3. Год выпуска;
4. Издательство
5. Обложка (картинка).

Приложение должно хранить данные в файле в виде (через разделители):

Название%Автор%1999%Издательство%1.jpg.

Приложение должно быть разделено визуально на 2 части: в верхней части должна быть таблица, а в нижней поля, через которые осуществляется добавление строки.
В приложении должно быть предусмотрено меню (загрузка из файла, сохранение, выход и т.д.).


Меню
Таблица
Поля
Кнопка добавления