

ВВЕДЕНИЕ

Важнейшую часть любой программы составляет ее пользовательский интерфейс. Собственно говоря, пользователя мало интересуют те сложные алгоритмы обработки данных и удачные находки, реализованные внутри программы, также мало его занимают и технологии, примененные для создания самого приложения и его пользовательского интерфейса. Пользователь видит только то, что видит, и, отталкиваясь от этого факта, и следует решать все задачи по проектированию и созданию пользовательского интерфейса. Интерфейс должен быть максимально удобным для пользователя и максимально быстрым; само собой, не последнее место занимает эстетическое удовлетворение, получаемое от работы с программой.

Все клиентские приложения системы «iBank 2 UA» реализованы на Java. Пользовательский интерфейс этих приложений создается с помощью библиотеки Swing, которая без всяких сомнений является ядром Java Foundation Classes (набор библиотек платформы J2SE)

Библиотека Swing содержит компоненты для создания пользовательского интерфейса, такие как таблицы, текстовые поля, надписи, переключателя и т.п., и инструменты для работы с этими компонентами. Библиотека Swing великолепно спланирована и реализована и способна дать все, что только может пожелать разработчик пользовательского интерфейса.

Опустим архитектуру и «глубины» Swing, поскольку нашей целью является только общее знакомство с принципами построения пользовательских интерфейсов в системе «iBank 2 UA». Руководство позволит вам получить знания об основных компонентах, которые используются в пользовательских интерфейсах системы «iBank 2 UA», познакомит с принципами размещения этих компонентов на форме. Данное руководство даст вам начальный практический навык создания пользовательских интерфейсов, за подробными же описаниями концепции и архитектуры Swing обращайтесь к дополнительной литературе.

Для кого предназначен данный документ?

Прежде всего, для сотрудников IT-подразделений банка, на чьи плечи ложится изменение предоставленного по умолчанию системой «iBank 2 UA» сценария регистрации клиентов.

Подразумевается, что вы имеете некоторый опыт в программировании и уже пробовали создавать пользовательские интерфейсы с помощью Java или *средств быстрой разработки* (Rapid Application Development, RAD), так что представленные в руководстве конструкции и термины, применяемые в программировании и создании пользовательских интерфейсов, не введут вас в заблуждение.

Структура руководства

Элементы управления. Элементы управления используются в интерфейсе повсюду, с их помощью пользователь влияет на ход выполнения программы. Арсенал предоставляемых Swing элементов управления велик и разнообразен, но пользоваться им просто, в этом вы убедитесь после прочтения данной главы.

Искусство расположения. В этой главе мы постараемся овладеть тайнами процесса расположения элементов на форме, используя менеджер расположения.

Примеры. Руководство содержит несколько примеров, призванных максимально ускорить и упростить ваше знакомство с аспектами работы элементов управления в Swing и менеджера расположения GridBagLayout. Примеры намеренно сделаны очень простыми и по возможности лаконичными: так они акцентируют ваше внимание на том, как что-то можно сделать, а не потрясают вас великолепным интерфейсом, который можно создать с помощью более сложного и объемного кода. Эти примеры станут для вас неплохим справочником, и после прочтения руководства: открыв страницу с нужным примером, вы мгновенно поймете, как работать с тем или иным компонентом Swing, как придать компонентам необходимое расположение на форме.

ЭЛЕМЕНТЫ УПРАВЛЕНИЯ

Взгляните на современные приложения с графическим интерфейсом и скажите, какие компоненты интерфейса неизменно присутствуют в каждом из них. Конечно, это элементы управления. В разных обликах и для разных целей, упакованные в меню, выстроенные в строгие ряды панелей инструментов, они вездесущи. Элементы управления позволяют пользователю общаться с программой и управлять ее ходом.

Swing содержит абсолютно полный арсенал придуманных на сегодняшний день элементов управления — это кнопки, флажки, переключатели, меню и его элементы и многое другое. Основные свойства, присущие любому элементу управления Swing, проще всего изучать на примере компонента.

Элементы управления Swing содержат огромное множество свойств, которые обеспечивают визуальное отображение элемента, его расположение и т.п. Компания «БИФИТ» использует только часть стандартных свойств для построения элементов управления, имена которых могут значительно отличаться от общепринятых.

Контейнер Panel

Контейнер Panel — это невидимый компонент графического интерфейса, служащий для объединения нескольких других компонентов в один объект типа Panel.

При добавлении компонента Panel указывается необходимый менеджер расположения, а он в свою очередь уже размещает компоненты в контейнере.

При размещении группы компонентов в контейнере Panel для перемещения всей группы компонентов достаточно переместить панель, и находящиеся на ней объекты автоматически переместятся вместе с ней, не изменив своего взаимного положения.

Для панелей Panel, основных строительных кирпичиков, в которых вы размещаете свои компоненты, в качестве визуального организующего элемента чаще всего используются Рамки (Border). В конечном итоге интерфейс складывается из множества определенным образом расположенных панелей Panel. Панели, содержащие часть элементов интерфейса, отвечающих за определенные функции, удобно «помещать в рамку». Благодаря таким рамкам пользователю проще освоиться с приложением, потому что он быстро определяет, какая часть интерфейса за что отвечает, и без дополнительных усилий находит то, что ему нужно.

Таблица 1. Параметры компонента Panel

Параметры	Предназначение
border	Тип окаймления, используемого при отображении панели, имеет свойство name - тип окаймления, может принимать значения ETCHED, SIMPLE, LOWERED, TITLED. В случае использования ETCHED, SIMPLE, LOWERED допустимо применять параметр enclose в котором указывается маска замыкания контура. В случае использования TITLED необходимо использовать параметр label, в котором содержится название метки, если эту метку необходимо использовать как link, то необходимо указание параметра action
layout	Менеджер расположения, используемый при раскладке компонентов внутри контейнера, имеет свойство name - название менеджера расположения, может принимать следующие значения: border, flow, grid, gridbag. Во всех примерах руководства далее подразумевается использование gridbag
var	Определяет переменную, которую в дальнейшем можно использовать в описании свойств. Имеет следующие свойства: name - название переменной, value - значение переменной.
constraints	Свойств нет, содержит элементы param для расположения компонентов в контейнере (см. Менеджер расположения GridBagLayout)

Разделитель Separator

Для упорядочивания информации в контейнере можно использовать разделители Separator. Это горизонтальные или вертикальные линии, разделяющие организованные в контейнере элементы управления.

Таблица 2. Свойства компонента Separator

Свойства	Предназначение
orientation	Ориентация: HORIZONTAL – горизонтальная, VERTICAL – вертикальная

Картинка Image

Иногда бывает необходимо привлечь внимание пользователя путем добавления какой-либо графической информации. Для этого служит компонент Image.

Таблица 3. Свойства компонента Image

Свойства	Предназначение
path	Путь к файлу с картинкой

Надписи Label и Textarealabel

Компонент-надпись Label используется для вывода текста, позволяя сообщить пользователю все, что вы захотите. Надо сказать, что надпись Label встречается в Swing буквально на каждом шагу: ее применяют для вывода вспомогательного текста в диалоговых окнах, для рисования значков и рисунков, и все сложные компоненты Swing, такие как деревья и таблицы, используют экземпляр надписи для прорисовки своих элементов: листьев деревьев и ячеек таблиц. Это настоящая «рабочая лошадка» библиотеки. Компонент Textarealabel используется для вывода многострочного текста.

Таблица 4. Свойства компонента Label

Свойства	Предназначение
text	Отображаемый текст
border	Признак использования рамки: Возможные значения: 0 – не использовать; 1 – использовать
align	Выравнивание текста: Возможные значения: LEFT – влево, CENTER – по центру, RIGHT – вправо

Таблица 5. Свойства компонента Textarealabel

Свойства	Предназначение
text	Отображаемый текст

Текстовые поля Textfield

Самые простые и наиболее часто встречающиеся в пользовательских интерфейсах текстовые компоненты. Как правило, такое поле является однострочным и служит для ввода текста. В библиотеке Swing имеется два текстовых поля. Первое, позволяет вводить некоторый простой однострочный текст (разнообразные атрибуты форматирования, такие как дополнительные цвета и шрифты, не поддерживаются). Второе поле дает возможность организовать ввод «секретной» информации (чаще всего паролей), которая не должна напрямую отображаться на экране.

Таблица 6. Свойства компонента Textfield

Свойства	Предназначение
cols	Количество столбцов (целое число)
editable	Возможность редактирования. Возможные значения: 0 – редактирование запрещено, 1 – редактирование разрешено

Многострочные поля Textarea

Многострочное текстовое поле Textarea представляет собой немного более расширенную версию обычного текстового поля Textfield. Как и обычное поле Textfield, многострочное поле Textarea предназначено для ввода простого неразмеченного различными атрибутами текста, но в отличие от обычных полей, позволяющих вводить только одну строку текста, многострочные поля дают пользователю возможность вводить произвольное количество строк текста. Кроме того, поле Textarea часто используется для вывода разного рода подробной информации о работе приложения, отображать которую в диалоговых окнах или в консоли неудобно. Вывод информации в многострочное текстовое поле позволяет пользователю без труда просмотреть ее, как бы много ее ни было, и при необходимости скопировать или изменить. Работают с многострочными полями так же, как с обычными, за тем исключением, что для них приходится задавать не только ширину (максимальное количество символов), но и высоту (максимальное количество строк).

Таблица 7. Свойства компонента Textarea

Свойства	Предназначение
cols	Количество столбцов (целое число)
rows	Количество строк (целое число)
editable	Возможность редактирования. Возможные значения: 0 – редактирование запрещено, 1 – редактирование разрешено
autoWrap	Включение переноса текста по строкам: Возможные значения: 0 - перенос не осуществляется, 1 – включен перенос по строкам

Кнопки Button

В простейшем виде кнопки — это самые обычные прямоугольники с текстом; пользователь щелкает на них мышью, чтобы выполнить какое-либо действие или о чем-либо просигнализировать.

Таблица 8. Свойства компонента Button

Свойства	Предназначение
text	Текст, отображаемый на кнопке

Флажки CheckBox и переключатели

Флажки используются там, где нужно предоставить пользователю возможность что-то включить или выключить. Они также группируются, но никогда не реализуют выбор «один из нескольких», а всегда позволяют выбрать несколько равноценных вариантов.

В подавляющем же большинстве ситуаций для реализации выбора «один из многих» применяются переключатели. По одиночке в интерфейсе они практически не встречаются, обычно на эту роль выбирают флажки.

Таблица 9. Свойства компонентов

Свойства	Предназначение
text	Текст, отображаемый рядом с флажком/радиокнопкой
group	Группа кнопки – если поле == null то это не связанная кнопка, иначе связанная

Однострочный список Choice

Раскрывающиеся списки Choice служат для выбора одного из множества доступных вариантов, примерно так же, как это происходит в группе переключателей RadioButton. В отличие от групп кнопок, которые располагаются на экране все вместе, в раскрывающемся списке Choice виден только сам выбранный элемент, а список возможных альтернатив появляется, только когда этого захочет пользователь (щелкнув на специальной кнопке раскрытия списка). Список возможных альтернатив выводится в специальном всплывающем меню, которое после выбора скрывается. Таким образом, раскрывающиеся списки (в отличие от групп элементов управления и обычных списков) позволяют экономить место в контейнере, даже если список альтернатив велик.

Таблицы Table

Таблица Table позволяет выводить двухмерные данные, записанные в виде строк и столбцов. Изменение свойств таблицы, при настройке шагов для регистрации клиентов, на данный момент невозможно.

ИСКУССТВО РАСПОЛОЖЕНИЯ

Язык Java недаром носит титул переносимого между платформами — со всеми различиями операционных систем прекрасно справится виртуальная машина Java. Однако остается вопрос правильного расположения компонентов на форме — здесь необходима гибкость и независимость от конкретных размеров. Все это обеспечивает *менеджер расположения* (layout manager), который играет очень важную роль в разработке пользовательского интерфейса. Данный компонент является невидимым элементом контейнера и берет на себя всю «грязную» работу по масштабированию и позиционированию видимых элементов (кнопки, поля ввода и др.) при изменении размеров контейнера. Использование менеджера расположения крайне важно, т.к. позволит вам гарантировать предсказуемый внешний вид вашей программы для различных режимов показа (развернутое на весь экран и т.д.) и разрешения монитора.

Как же работает менеджер расположения? Контейнер вызывает методы менеджера расположения каждый раз при изменении своих размеров или при первом появлении на экране. Менеджер расположения обязан расположить добавляемые в контейнер компоненты в некотором порядке, зависящем от реализованного в нем алгоритма, и придать им некоторый размер.

В стандартной библиотеке Java существует несколько готовых менеджеров расположения, и с их помощью можно реализовать абсолютно любое расположение. В пакете разработки JDK 1.4 имеется 7 готовых универсальных менеджеров расположения. Условно и можно разделить на несколько групп: очень простые (к ним относятся FlowLayout, GridLayout и BorderLayout), очень сложные (к таким без всяких сомнений стоит отнести расположение GridBagLayout) и специализированные (CardLayout и SpringLayout). Не было перечислено только расположение BoxLayout, потому что оно может быть и очень простым (но при этом мощным), и достаточно сложным, а при надобности и специализированным. «Пройдемся» от простого к сложному.

Менеджер **BorderLayout** специально предназначен для обычных и диалоговых окон, потому что позволяет быстро и просто расположить наиболее часто используемые элементы любого окна: панель инструментов, строку состояния и основное содержимое. Для этого окно разбивается им на четыре области, или полюса (отсюда его название), а все оставшееся место заполняется компонентом, выполняющим основную функцию приложения (например, в редакторе это будет текстовое поле, в многооконном приложении — рабочий стол).

Последовательное расположение **FlowLayout** работает очень просто, но, тем не менее, эффективно. Оно выкладывает компоненты в контейнер, как пирожки на противень: слева направо, сверху вниз.

Табличное расположение **GridLayout**. Как нетрудно догадаться по названию, менеджер расположения GridLayout разделяет контейнер на таблицу с ячейками одинакового размера. Количество строк и столбцов можно указать в конструкторе, причем существует возможность задать произвольное количество либо строк, либо столбцов (но не одновременно). Все ячейки имеют одинаковый размер, равный размеру самого большого компонента, находящегося в таблице. Табличное расположение идеально подходит для ситуаций, где нужно разбить контейнер на несколько одинаковых частей с примерно одинаковым по размеру содержимым.

Менеджер размещения **CardLayout** своеобразен — он показывает в контейнере только один, первый, компонент. Остальные компоненты лежат под первым в определенном порядке как игральные карты в колоде.

Блочное расположение **BoxLayout** — прекрасная альтернатива всем остальным менеджерам расположения. Обладая возможностями GridBagLayout, расположение BoxLayout не сложнее BorderLayout. Менеджер блочного расположения выкладывает компоненты в контейнер блоками:

столбиком (по оси Y) или полоской (по оси X), при этом каждый отдельный компонент можно выравнивать по центру, по левому или по правому краям, а также по верху или по низу.

И в последнюю очередь рассмотрим менеджер расположения GridBagLayout.

Именно этот менеджер лежит в основе построения графических форм компанией «БИФИТ» ввиду своей универсальности и широких возможностей.

Менеджер расположения GridBagLayout

Принцип работы менеджера GridBagLayout прост: контейнер разбивается на сетку, и над каждой ячейкой сетки у нас имеется неограниченный контроль. Компоненты могут занимать любое количество ячеек сетки, ячейки могут оставаться пустыми, одним словом, способов управления множество.

Перечислим поля, с помощью которых GridBagLayout располагает компоненты на форме, и дадим их краткое описание.

gridx и gridy

Поля gridx и gridy задают, соответственно, номер столбца и номер строки для ячейки, в которую будет помещен компонент. Верхней левой ячейке соответствуют нулевые значения.

gridwidth и gridheight

Поля gridwidth и gridheight определяют количество ячеек, занимаемых добавляемым компонентом.

Если компонент полностью помещается в одну ячейку, вы можете задать в этих полях значение единицы. Если же компонент должен занимать, например, две смежные ячейки в одной строке, то для gridwidth нужно задать значение, равное двум, а для gridheight - значение, равное единице.

fill

Поле fill определяет стратегию распределения компоненту свободного пространства ячейки или ячеек таблицы, если размеры компонента меньше размеров выделенного для него места.

Возможны следующие значения:

NONE - Компонент не изменяет своих размеров

BOTH - Изменяется высота и ширина, причем таким образом, чтобы компонент занимал все отведенное для него пространство

HORIZONTAL - Компонент растягивается по горизонтали

VERTICAL - Компонент растягивается по вертикали

anchor

Поле anchor задает выравнивание компонента внутри отведенного для него пространства. Он включается в работу, когда размеры компонента меньше размеров выделенного для него места.

Для поля anchor вы можете указать следующие значения:

CENTER – Центрирование

NORTH – Север

SOUTH – Юг

EAST – Восток

WEST – Запад

NORHEAST – Северо-восток

NORTHWEST – Северо-запад

SOUTHEAST – Юго-восток

SOUTHWEST – Юго-запад

weightx и weighty

Эти поля определяют стратегию изменения размеров компонента, отвечая за выделение пространства для столбцов (`weightx`) и строк (`weighty`).

Если записать в них нулевые значения все добавленные компоненты соберутся в центре контейнера и будут выровнены по центру (как по вертикали, так и по горизонтали).

Чтобы размеры компонента изменялись по горизонтали или вертикали, в поля `weightx` и `weighty` нужно записать значения от 0.0 до 1.0.

Если в столбце несколько компонентов, то его ширина будет определяться компонентом с максимальным значением `weightx`. Аналогичное утверждение верно и для строк.

Заметим, что дополнительное пространство добавляется к строкам и столбцам снизу и справа, соответственно.

ipadx и ipady

С помощью полей `ipadx` и `ipady` вы можете указать, что размеры компонента необходимо увеличить на заданное количество пикселей, соответственно, по горизонтали и вертикали.

insets

Поле `insets` позволяет задать для компонента отступы от краев выделенной ему области (сверху, слева, снизу, справа). По умолчанию такие отступы отсутствуют.

ПРИМЕРЫ

А теперь применим все, что прочитали в предыдущих главах на практике и вы убедитесь, что в размещении компонентов и придании им желаемого вида нет ничего сложного.

Пример 1. Стандартное поведение GridBagLayout

Устанавливаем тип менеджера расположения как GridBagLayout и добавляем на него компоненты. Результатом будет «строка» компонентов, расположенных вплотную друг к другу в центре контейнера, каждый из компонентов будет иметь предпочтительный размер.

```
<form>
  <panel>
    <layout NAME="gridbag"/>
    <button ID="1" TEXT="Wonderful" MODE="2"/>
    <button ID="2" TEXT="World" MODE="2"/>
    <button ID="3" TEXT="Of" MODE="2"/>
    <button ID="4" TEXT="Swing!!!" MODE="2"/>
  </panel>
</form>
```

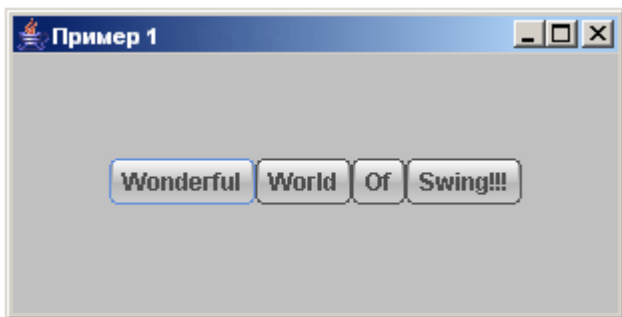


Рис.1. Стандартное поведение GridBagLayout

Ограничения (GridBagConstraints)

Устанавливая атрибуты GridBagConstraints, мы можем гибко управлять размещением компонентов в контейнере. Ниже приведены различные атрибуты и их значения по умолчанию. Поведение этих атрибутов рассмотрим в примере ниже.

Таблица 10. Значения по умолчанию для атрибутов GridBagConstraints

Название	Значение по умолчанию
gridx	0
gridy	0
gridwidth	1
gridheight	1
weightx	0.0
weighty	0.0
anchor	CENTER
fill	NONE
insets	(0, 0, 0, 0)
ipadx	0
ipady	0

Пример 2. Использование gridx, gridy, insets, ipadx и ipady

В этом примере мы разместим кнопки “Wonderful” and “World” в первой строке, а другие кнопки на второй строке. Мы также установим insets, чтоб кнопки не выглядели беспорядочно и установим высоту и длину кнопок.

```

<form>
  <panel>
    <layout NAME="gridbag" />

    <button ID="1" TEXT="Wonderful" MODE="2">
    <constraints>
      <param name="gridx" value="0" />
      <param name="gridy" value="0" />
      <param name="ipadx" value="5" />
      <param name="ipady" value="5" />
      <param NAME="insets" VALUE="2 2 2 2" />
    </constraints>
  </button>

    <button ID="2" TEXT="World" MODE="2">
    <constraints>
      <param name="gridx" value="1" />
      <param name="ipadx" value="0" />
      <param name="ipady" value="0" />
      <param NAME="insets" VALUE="2 2 2 2" />
    </constraints>
  </button>

    <button ID="3" TEXT="Of" MODE="2">
    <constraints>
      <param name="gridx" value="0" />
      <param name="gridy" value="1" />
      <param NAME="insets" VALUE="2 2 2 2" />
    </constraints>
  </button>

    <button ID="4" TEXT="Swing!!!" MODE="2">
    <constraints>
      <param name="gridx" value="1" />
      <param NAME="insets" VALUE="2 2 2 2" />
    </constraints>
  </button>
  </panel>
</form>

```

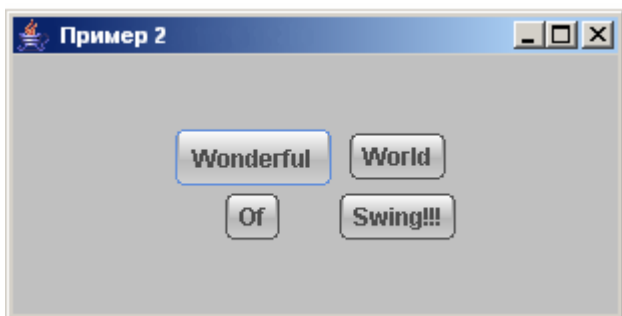


Рис.2. Использование gridx, gridy, insets, ipadx и ipady

Пример 3. Использование weightx и weighty

Сейчас мы изменим все ячейки кнопок для того, чтобы заполнить все дополнительное пространство контейнера равномерно при изменении размеров контейнера. Установим `weightx=1.0` и `weighty=1.0` и оставим эти атрибуты неизменными для каждого компонента, скажем менеджеру расположения использовать все доступное пространство для каждой ячейки.

```

<form>
  <panel>
    <layout NAME="gridbag" />

    <button ID="1" TEXT="Wonderful" MODE="2">
    <constraints>
      <param NAME="insets" VALUE="2 2 2 2" />
      <param name="gridx" value="0" />

```

```

        <param name="gridy" value="0"/>
        <param name="weightx" value="1.0"/>
        <param name="weighty" value="1.0"/>
    </constraints>
</button>

<button ID="2" TEXT="World" MODE="2">
<constraints>
    <param NAME="insets" VALUE="2 2 2 2"/>
    <param name="gridx" value="1"/>
    <param name="gridy" value="0"/>
    <param name="weightx" value="1.0"/>
    <param name="weighty" value="1.0"/>
</constraints>
</button>

<button ID="3" TEXT="Of" MODE="2">
<constraints>
    <param NAME="insets" VALUE="2 2 2 2"/>
    <param name="gridx" value="0"/>
    <param name="gridy" value="1"/>
    <param name="weightx" value="1.0"/>
    <param name="weighty" value="1.0"/>
</constraints>
</button>

<button ID="4" TEXT="Swing!!!" MODE="2">
<constraints>
    <param NAME="insets" VALUE="2 2 2 2"/>
    <param name="gridx" value="1"/>
    <param name="weightx" value="1.0"/>
    <param name="weighty" value="1.0"/>
</constraints>
</button>
</panel>
</form>

```

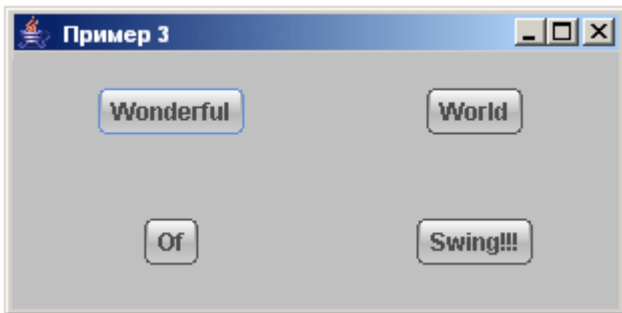


Рис. 3. Использование weightx и weighty

Пример 4. Использование gridwidth и gridheight

GridbagLayout также позволяет нам разнести компоненты вдоль нескольких ячеек, используя gridwidth и gridheight. Чтобы это продемонстрировать, изменим наш пример так, чтобы заставить кнопку «Wonderful» занять 2 строчки, а кнопку «World» занять 2 столбца.

```

<form>
  <panel>
    <layout NAME="gridbag"/>

    <button ID="1" TEXT="Wonderful" MODE="2">
      <constraints>
        <param NAME="insets" VALUE="2 2 2 2"/>
        <param name="gridx" value="0"/>
        <param name="gridy" value="0"/>
        <param name="weightx" value="1.0"/>
        <param name="weighty" value="1.0"/>
        <param name="gridheight" value="2"/>
      </constraints>
    </button>

```

```

<button ID="2" TEXT="World" MODE="2">
<constraints>
  <param NAME="insets" VALUE="2 2 2 2"/>
  <param name="gridx" value="1"/>
  <param name="gridy" value="0"/>
  <param name="weightx" value="1.0"/>
  <param name="weighty" value="1.0"/>
  <param name="gridheight" value="1"/>
  <param name="gridwidth" value="2"/>
</constraints>
</button>

<button ID="3" TEXT="Of" MODE="2">
<constraints>
  <param NAME="insets" VALUE="2 2 2 2"/>
  <param name="gridx" value="1"/>
  <param name="gridy" value="1"/>
  <param name="weightx" value="1.0"/>
  <param name="weighty" value="1.0"/>
  <param name="gridheight" value="1"/>
  <param name="gridwidth" value="1"/>
</constraints>
</button>

<button ID="4" TEXT="Swing!!!" MODE="2">
<constraints>
  <param NAME="insets" VALUE="2 2 2 2"/>
  <param name="gridx" value="2"/>
  <param name="gridy" value="1"/>
  <param name="weightx" value="1.0"/>
  <param name="weighty" value="1.0"/>
  <param name="gridheight" value="1"/>
  <param name="gridwidth" value="1"/>
</constraints>
</button>
</panel>
</form>

```

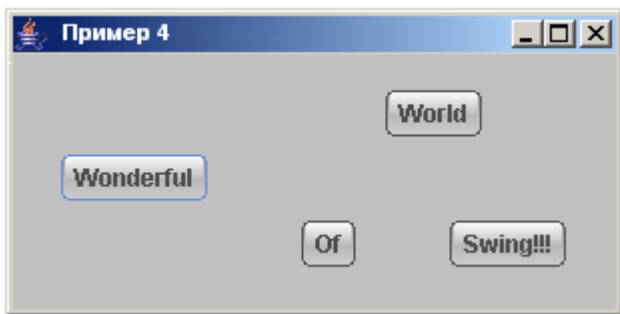


Рис.4. Использование gridwidth и gridheight

Пример 5. Использование anchor

Мы можем контролировать, как располагается компонент внутри ячейки, задавая anchor. Далее мы изменим наш пример так, чтобы кнопка «Wonderful» располагалась на СЕВЕР, кнопка «World» на ЮГО-ЗАПАД, а кнопки «Of» и «Swing!!!» в ЦЕНТР.

```

<form>
  <panel>
    <layout NAME="gridbag"/>

    <button ID="1" TEXT="Wonderful" MODE="2">
      <constraints>
        <param NAME="insets" VALUE="2 2 2 2"/>
        <param name="gridx" value="0"/>
        <param name="gridy" value="0"/>
        <param name="weightx" value="1.0"/>
        <param name="weighty" value="1.0"/>
        <param name="gridheight" value="2"/>
        <param name="anchor" value="NORTH"/>
      </constraints>

```

```

</button>

<button ID="2" TEXT="World" MODE="2">
<constraints>
  <param NAME="insets" VALUE="2 2 2 2"/>
  <param name="gridx" value="1"/>
  <param name="gridy" value="0"/>
  <param name="weightx" value="1.0"/>
  <param name="weighty" value="1.0"/>
  <param name="gridheight" value="1"/>
  <param name="gridwidth" value="2"/>
  <param name="anchor" value="SOUTHWEST"/>
</constraints>
</button>

<button ID="3" TEXT="Of" MODE="2">
<constraints>
  <param NAME="insets" VALUE="2 2 2 2"/>
  <param name="gridx" value="1"/>
  <param name="gridy" value="1"/>
  <param name="weightx" value="1.0"/>
  <param name="weighty" value="1.0"/>
  <param name="gridheight" value="1"/>
  <param name="gridwidth" value="1"/>
  <param name="anchor" value="CENTER"/>
</constraints>
</button>

<button ID="4" TEXT="Swing!!!" MODE="2">
<constraints>
  <param NAME="insets" VALUE="2 2 2 2"/>
  <param name="gridx" value="2"/>
  <param name="gridy" value="1"/>
  <param name="weightx" value="1.0"/>
  <param name="weighty" value="1.0"/>
  <param name="gridheight" value="1"/>
  <param name="gridwidth" value="1"/>
</constraints>
</button>
</panel>
</form>

```



Рис. 5. Использование anchor

Пример 6. Использование fill

Изменим наш пример, чтобы заставить кнопку «Wonderful» занять все доступное пространство ячейки и по горизонтали и по вертикали. Кнопка «World» займет все доступное пространство ячейки по горизонтали, а по вертикали будет иметь свой предпочтительный размер. Для кнопки «Of» не будем использовать fill, она будет иметь свой предпочтительный размер. Кнопка «Swing!!!» займет все доступное пространство ячейки по вертикали, а по горизонтали будет иметь свой предпочтительный размер.

```

<form>
  <panel>
    <layout NAME="gridbag"/>

    <button ID="1" TEXT="Wonderful" MODE="2">
      <constraints>

```

```

        <param NAME="insets" VALUE="2 2 2 2"/>
        <param name="gridx" value="0"/>
        <param name="gridy" value="0"/>
        <param name="weightx" value="1.0"/>
        <param name="weighty" value="1.0"/>
        <param name="gridheight" value="2"/>
        <param name="fill" value="BOTH"/>
    </constraints>
</button>

<button ID="2" TEXT="World" MODE="2">
<constraints>
    <param NAME="insets" VALUE="2 2 2 2"/>
    <param name="gridx" value="1"/>
    <param name="gridy" value="0"/>
    <param name="weightx" value="1.0"/>
    <param name="weighty" value="1.0"/>
    <param name="gridheight" value="1"/>
    <param name="gridwidth" value="2"/>
    <param name="fill" value="HORIZONTAL"/>
</constraints>
</button>

<button ID="3" TEXT="Of" MODE="2">
<constraints>
    <param NAME="insets" VALUE="2 2 2 2"/>
    <param name="gridx" value="1"/>
    <param name="gridy" value="1"/>
    <param name="weightx" value="1.0"/>
    <param name="weighty" value="1.0"/>
    <param name="gridheight" value="1"/>
    <param name="gridwidth" value="1"/>
    <param name="fill" value="NONE"/>
</constraints>
</button>

<button ID="4" TEXT="Swing!!!" MODE="2">
<constraints>
    <param NAME="insets" VALUE="2 2 2 2"/>
    <param name="gridx" value="2"/>
    <param name="gridy" value="1"/>
    <param name="weightx" value="1.0"/>
    <param name="weighty" value="1.0"/>
    <param name="gridheight" value="1"/>
    <param name="gridwidth" value="1"/>
    <param name="fill" value="VERTICAL"/>
</constraints>
</button>
</panel>
</form>

```



Рис. 6. Использование fill

ЗАКЛЮЧЕНИЕ

В руководстве использованы материалы:

1. Иван Портянкин «Swing. Эффективные пользовательские интерфейсы», Санкт-Петербург 2005 г.
2. Matthew Robinson, Pavel Vorobiev «Swing»
3. <http://java.sun.com/>