|  |  |
| --- | --- |
| ***Таблица 2.1.*** *Важнейшие свойства, которые можно устанавливать для компонентов* | |
| **Название свойства** | **Что оно задает** |
| action | Объект типа javax.swing.Action, выполняющий действие при наступлении события actionEvent. В NetBeans 6.5 осуществлен переход от обработчиков событий ActionPerformed к обработ-чикам типа Action (про акции *см. главу 13*) |
| background | Цвет фона |
| font | Шрифт надписи на компоненте |
| foreground | Цвет надписи на компоненте |
| icon | Пиктограмма (небольшая картинка, иконка), которая рисуется на компоненте рядом с текстом. Устанавливается ссылка на объект типа Icon (*см. разд. 2.11, 13.3—13.4* и *13.7*) |
| mnemonic | Код клавиши-акселератора на клавиатуре (например, при вво-де в это поле символа q наша кнопка активизируется и нажи-мается по комбинации <Alt>+<q>) |
| text | Текст (надпись) на компоненте |
| toolTipText | Всплывающая подсказка, появляющаяся через некоторое вре-мя при наведении курсора мыши на компонент |
| **Категория Other Properties** (другие свойства) | |
| UIClassID | Строка, задающая имя класса, определяющего внешний вид компонента в технологии Look and Fill *(см. разд. 2.9)* |
| actionCommand | Строка, посылаемая в акцию-обработчик события при наступ-лении соответствующего события, например, нажатия кнопки *(см. главу 13)*. Обработчик может выполнять те или иные дей-ствия в зависимости от переданной строки |
| alignmentX | Предпочтительное горизонтальное смещение компонента |
| alignmentY | Предпочтительное вертикальное смещение компонента |
| autoscrolls | Наличие скроллеров — полос прокрутки, возникающих, когда видимый размер компонента меньше, чем нужный для вывода текста или других элементов этого компонента |
| baselineResizeBehavior | Перечисление *(см. разд. 3.8)*, указывающее, как должна ме-няться базовая линия компонента при изменении его размера (обычно базовая линия проходит по центру компонента и отно-сительно нее выравнивается текст и изображения) |
| border | Тип рамки вокруг компонента |
| borderPainted | Рисуется ли рамка вокруг компонента |
| buttonGroup | Позволяет объединять кнопки в группы с помощью назначения этому полю имени компонента типа ButtonGroup. Важно для объединения в группы кнопок типа ToggleButton и RadioButton |
| componentPopupMenu | Позволяет назначать контекстное меню, появляющееся по на-жатию правой кнопкой мыши в области компонента |
| contentAreaFilled | Указывает, имеется ли заполнение цветом внутренней области компонента (для кнопок оно создает эффект трехмерности, без заполнения кнопка выглядит плоской) |
| defaultCapable | Указывает, способна ли кнопка быть кнопкой по умолчанию: при нажатии клавиши <Enter> автоматически происходит нажа-тие кнопки по умолчанию (такая кнопка на экранной форме должна быть единственной) |
| disabledIcon | Картинка, которая рисуется на компоненте рядом с текстом в случае, когда компонент недоступен (disabled) (т. е. когда его свойство enabled установлено в false) |
| disabledSelectedIcon | Картинка, которая рисуется на компоненте рядом с текстом в случае, когда выделенный компонент недоступен (disabled) (т. е. когда его свойство enabled установлено в false) |
| displayedMnemonicIndex | Код символа-акселератора, назначенного компоненту (т. е. соответствующее этому символу число типа char) |
| doubleBuffered | Обеспечивает режим двойной буферизации для отрисовки на экране при установке флажка для данного свойства (оно при этом приобретает значение true). Этот режим следует зада-вать только для компонентов, на которых будет проводиться вывод динамических графических данных. Двойная буфериза-ция устраняет мерцание при динамическом выводе графики, но потребляет в два раза больше ресурсов |
| enabled | Указывает, доступен ли компонент. По умолчанию все созда-ваемые на форме компоненты доступны. Недоступные компо-ненты рисуются более блеклыми красками и на них нельзя установить фокус (т. е. перейти на них мышью или с помощью горячих клавиш) |
| focusCycledRoot | Указывает, является ли компонент-контейнер корневым для цикла навигации при нажатии на клавиатуре клавиш навигации. Если компонент корневой, навигация осуществляется только внутри него между дочерними компонентами. См. также далее информацию о свойстве nextFocusableComponent |
| focusTraversalPolicy | Задает способ перемещения фокуса от компонента к компо-ненту (навигации) при нажатии на клавиатуре клавиш навига-ции. Способ зависит от того, ссылка на объект какого типа назначена данному свойству: ContainerOrderFocusTraversalPolicy или DefaultFocusTraversalPolicy — по порядку компонентов в контейнере; SortingFocusTraversalPolicy — сортирует компоненты с помощью метода-компаратора; LayoutFocusTraversalPolicy — сортирует компоненты в зави-симости от их положения и размера. См. также далее инфор-мацию о свойстве nextFocusableComponent |
| focusTraversalPolicyProvider | Булево свойство, указывающее, что свойство focusTraversalPolicy именно этого компонента-контейнера будет использовано в качестве провайдера политики навигации |
| focusPainted | Указывает на необходимость перерисовки компонента при получении им фокуса. Если пользовательские средства отри-совки компонента не используются, то ни на что не влияет |
| focusable | Указывает, способен ли компонент захватывать фокус. Свойст-во важно для компонентов редактирования текста и компонен-тов-контейнеров |
| hideActionText | Булево свойство, предназначенное для управления компонен-том, которому назначена акция *(см. главу 13)*. Если значение равно false (по умолчанию), при выполнении акции компонент начинает отображать текст, переданный в акцию (свойство actionCommand). В противном случае этот текст не отобража- ется |
| horizontalAlignment | Горизонтальное выравнивание текста и/или картинки, нарисо-ванной на компоненте.  Имеет значения:  CENTER (по центру — значение по умолчанию);  LEADING (ведущий, лидирующий);  LEFT (по левому краю);  RIGHT (по правому краю);  TRAILING ("прицепленный сзади").  Для горизонтально пишущегося текста разницы между LEADING и LEFT, а также между RIGHT и TRAILING нет |
| horizontalTextPosition | Горизонтальное положение текста по отношению к картинке, нарисованной на компоненте.  Имеет значения:  CENTER (картинка по центру текста);  LEADING, LEFT (текст слева от картинки);  RIGHT, TRAILING (текст справа от картинки) |
| verticalAlignment | Вертикальное выравнивание текста и/или картинки, нарисован-ной на компоненте.  Имеет значения:  CENTER (по центру);  TOP (расположение по верху компонента);  BOTTOM (расположение по низу компонента) |
| verticalTextPosition | Вертикальное выравнивание текста по отношению к картинке, нарисованной на компоненте.  Имеет значения:  CENTER (картинка по центру текста);  TOP (картинка по верху текста);  BOTTOM (картинка по низу текста) |
| iconTextGap | Расстояние (зазор) в пикселах (точках экрана) между картин-кой, нарисованной на компоненте (если она есть), и текстом, написанном на компоненте |
| inheritsPopupMenu | Наследует контекстное меню. Если установить это поле в true (установить флажок в редакторе свойств), то компонент унас-ледует контекстное меню от того группирующего компонента (контейнера), на котором он расположен |
| inputVerifier | Ссылка на объект типа javax.swing.InputVerifier, предназна-ченный для проверки правильности ввода текста в компонент |
| insets | Задает для компонента-контейнера отступы сверху, слева, сни-зу и справа, которые не должны быть заняты дочерними ком-понентами |
| label | Практически то же, что и Text, но имеется только для компо-нентов с неизменяемым текстом (реально это ссылка на объект типа java.awt.Label). |
| maximumSize | Максимально допустимый размер компонента — в случае, ко-гда его размер может изменяться во время работы программы. Важен для форм, у которых свойство resizable установлено в true |
| minimumSize | Минимально допустимый размер компонента (в случае, когда его размер может изменяться во время работы программы). Важен для форм, у которых свойство resizable установлено в true |
| model | Ссылка на объект модели представления данных для компо-нента. Обычно создание специальных объектов-моделей необ-ходимо только для таблиц, и то в NetBeans имеются средства создания таких объектов автоматически — в режиме визуаль-ного проектирования таблицы |
| multiclickThreshold | Задает для компонента порог (в миллисекундах) для двойного щелчка мыши |
| name | Строка с именем компонента. Это просто строковое значение, не имеющее ничего общего с именем компонента в Java-коде |
| nextFocusableComponent | Ссылка на компонент, который должен быть следующим при нажатии на клавишу навигации <Tab> на клавиатуре. По умол-чанию компонентам могут быть не назначены такие ссылки, и навигация будет отсутствовать. Это зависит от стиля LookAndFeel — *см. разд. 2.9*. При стиле System под Windows навигация присутствует, при стиле Metal — отсутствует |
| opaque | Непрозрачный. При снятии флажка, установленного по умолча-нию, компонент делается прозрачным (эта возможность появи-лась только начиная с Windows 2000). Не следует пользоваться прозрачностью без особой необходимости — она занимает во время работы программы много ресурсов процессора |
| paintingForPrint | Булево свойство. Если задать его значение равным true (установить флажок), то при печати родительского компонента-контейнера методом print будет осуществлена печать данного дочернего компонента (вызван его метод print) |
| preferredSize | Предпочтительный размер компонента (задается размер в пикселах по горизонтали и по вертикали) |
| pressedIcon | Картинка, которая рисуется на компоненте рядом с текстом в случае, когда кнопка нажата (свойство относится только к кнопкам) |
| requestFocusEnabled | Булево свойство, задающее способность компонента захваты-вать фокус после щелчка по нему мышью. Не влияет на захват фокуса при навигации с помощью клавиатуры |
| rolloverEnabled | Булево свойство, задающее способность компонента менять цвет при наведении на него курсора мыши |
| rolloverIcon | Картинка, которая рисуется на компоненте рядом с текстом при наведении на компонент курсора мыши |
| rolloverSelectedIcon | Картинка, которая рисуется на выделенном компоненте рядом с текстом при наведении на этот компонент курсора мыши. Дан-ное свойство наиболее важно для пунктов меню и элементов списков |
| selected | Булево свойство. Если задать его значение равным true (уста-новить флажок), компонент будет отмечен как выделенный. Данное свойство наиболее важно для радиокнопок, элементов независимого выбора (JCheckBox), кнопок с фиксацией (JToggleButton), пунктов меню, элементов списков |
| selectedIcon | Картинка, которая рисуется на выделенном компоненте рядом с текстом |
| verifyInputWhenFocusTarget | Булево свойство, задающее необходимость проверки компо-нентом введенного в него текста при получении компонентом фокуса. Если компоненту не назначен объект inputVerifier, данное свойство ни на что не влияет |

|  |  |
| --- | --- |
| ***Таблица 2.2.*** *Дополнительные свойства экранных форм* | |
| **Название свойства** | **Что оно задает** |
| defaultCloseOperation | Задает операцию, которая выполняется по умолчанию при закрытии формы. По умолчанию это HIDE (сделать форму невидимой, но не удалять). Также возможны значения EXIT\_ON\_CLOSE (выход из прило-жения после закрытия формы), DO\_NOTHING (ничего не делать), DISPOSE (удаление формы после закрытия) |
| title | Задает заголовок формы |
| **Категория Other Properties** (другие свойства) | |
| alwaysOnTop | Булево свойство, задающее способность формы всегда находиться поверх других форм в раскрытом окне операционной системы |
| alwaysOnTopSupported | Булево свойство, доступное только для чтения, показывающее, под-держивает ли операционная система управление формами с помощью свойства alwaysOnTop |
| bounds | Размер формы задается пиксельными координатами левого верхнего угла и правого нижнего угла формы. Пиксел (светящаяся точка экра-на), соответствующий верхнему левому краю экрана, имеет координа-ты [0,0]. Следующий пиксел по горизонтали имеет координату [1,0], а по вертикали [0,1].  Форма, расположенная в левом верхнем углу и имеющая ширину 200 пикселов и высоту 100 пикселов, будет иметь границы [0,0,199,99]. Значение bounds игнорируется в случае, если задаваемые им размеры выходят за пределы, определяемые свойствами minimumSize или maximumSize |
| cursor | Задает вид курсора мыши при его нахождении над любыми компонен-тами формы. Например, при установке значения Crosshair Cursor он будет иметь вид вертикального крестика. Обычно значение этого свойства меняют динамически во время работы программы. Чаще всего — на Hand Cursor (в виде руки), обозначая доступность какого-либо действия с компонентом, над которым находится курсор, или на Wait Cursor — ожидание в виде песочных часов |
| locationByPlatform | Когда отсутствует программная установка положения формы, а свой-ство имеет значение true (по умолчанию false), то при показе форма устанавливается в позицию, задаваемую операционной системой. Особенно хорошо видно влияние этого свойства в случае, когда не-сколько раз запустили одно и то же приложение. Например, Windows располагает окна приложений с некоторым сдвигом друг относительно друга, чтобы было видно несколько запущенных программ |
| sizable | Позволяет разрешать (по умолчанию) или запрещать пользователю во время работы программы изменять мышью размер формы |
| state | Задает состояние окна: минимизированное (значение 1) или нормаль-ное (значение 0 или любое другое, отличное от 1) |

|  |  |  |
| --- | --- | --- |
| ***Таблица 2.3.*** *Компоненты библиотеки Swing* | | |
| **№** | **Компонент** | **Назначение компонента** |
| **Swing Controls** (управляющие элементы) | | |
| 1 | JLabel | Метка — вывод однострочного неформатированного текста |
| 2 | JButton | Кнопка — кнопка с текстом и/или с картинкой |
| 3 | JToggleButton | Защелкивающаяся кнопка — кнопка с фиксацией. Может быть одной из нескольких таких кнопок в группе, в этом случае нажа-тие одной кнопки вызывает отпускание другой. Работа группы обеспечивается компонентом ButtonGroup, который должен быть перетащен на форму, а затем назначен свойству buttonGroup |
| 4 | JCheckBox | Пункт выбора с независимой фиксацией |
| 5 | JRadioButton | Радиокнопка — пункт выбора с зависимой фиксацией. Должен быть одним из нескольких в группе. Работа группы обеспечива-ется компонентом ButtonGroup |
| 6 | ButtonGroup | Обеспечивает работу групп компонентов JToggleButton или JradioButton |
| 7 | JComboBox | Выпадающий список |
| 8 | JList | Прокручивающийся список |
| 9 | JTextField | Текстовое поле — однострочный пункт ввода и редактирования текста |
| 10 | JTextArea | Текстовая область — многострочный пункт ввода и редактирова-ния текста |
| 11 | JScrollBar | Независимая полоса прокрутки. Используется редко — для про-граммно-управляемой прокрутки содержимого компонентов, не имеющих встроенных полос прокрутки |
| 12 | JSlider | Ползунок. Служит для плавной регулировки числовых величин, а также связанных с ними программно-регулируемых изменений |
| 13 | JProgressBar | Полоса показа доли выполнения задачи. Показывает уровень, отражающий долю выполнения задачи |
| 14 | JFormattedTextField | Поле ввода форматированного текста |
| 15 | JPasswordField | Поле ввода пароля — вводимый текст отображается звездочками |
| 16 | JSpinner | Поле ввода числа с кнопками увеличения/уменьшения |
| 17 | JSeparator | Разделительная линия. Используется в декоративных целях для разделения рабочих областей формы и других группирующих компонентов |
| 18 | JTextPane | Текстовая панель. По умолчанию автоматически переносит текст на новую строку, а не располагает его в одну строку с показом горизонтального скроллера, как это делает JTextArea |
| 19 | JEditorPane | Панель текстового редактора |
| 20 | JTree | Дерево — показывает дерево, в котором каждая ветвь может быть с иконками и текстом, а узлы разворачиваются и сворачи-ваются |
| 21 | JTable | Таблица — показ текстовой таблицы. Имеет возможность запол-нения значениями по умолчанию на этапе проектирования |
| **Swing Menus** (элементы меню) | | |
| 1 | JMenuBar | Меню формы — предназначено для расположения в нем компо-нентов типа JMenu (заголовков меню), показываемых в меню |
| 2 | JMenu | Заголовок меню (в него можно добавлять пункты меню JMenuItem, JCheckBoxMenuItem и JradioButtonMenuItem) |
| 3 | JMenuItem | Пункт меню. Может добавляться к заголовкам меню JMenu и в контекстное меню JPopupMenu |
| 4 | JCheckBoxMenuItem | Пункт меню с элементом CheckBox — чтобы этот пункт меню можно было отмечать флажком. Может добавляться к заголов-кам меню JMenu и в контекстное меню JPopupMenu |
| 5 | JRadioButtonMenuItem | Пункт меню с радиокнопкой (чтобы этот пункт меню можно было отмечать точкой в кружке). Может добавляться к заголовкам ме-ню JMenu и в контекстное меню JPopupMenu |
| 6 | JPopupMenu | Контекстное меню — предназначено для показа по нажатию пра-вой кнопки мыши в области компонента, которому назначено свойство componentPopupMenu. Пункты меню состоят из компонентов типа: JMenuItem, JCheckBoxMenuItem и JRadioButtonMenuItem |
| 7 | JSeparator | Разделитель. Добавляется в меню для группировки пунктов ме-ню (разделяя их горизонтальной линией) |
| **Swing Containers** (контейнеры) | | |
| 1 | JPanel | Панель — группирующий компонент, позволяющий располагать на себе другие компоненты. Передвижение панели перемещает вместе с ней все расположенные на ней компоненты. По умолча-нию свойство layout (размещение) установлено как FlowLayout (в виде потока). Для простых задач вместо этого компонента лучше использовать JLayeredPane |
| 2 | JTabbedPane | Панель с вкладками — каждый положенный на нее компонент показывается в отдельной вкладке. Чтобы разместить на одной вкладке несколько компонентов, сначала положите на панель с вкладками обычную панель. Для того чтобы создать после-дующие вкладки, выделите панель с вкладками, вызовите правой кнопкой мыши контекстное меню, запустите команду **Add From Palette** (добавить из палитры) и добавьте еще одну панель или другой компонент |
| 3 | JSplitPane | Панель с разделителем — панель, состоящая из двух частей, между ними имеется линия разделителя, которую можно пере-таскивать мышью, меняя взаимный размер частей |
| 4 | JScrollPane | Панель с полосами прокрутки |
| 5 | JToolBar | Панель инструментов. Обычно на ней размещают кнопки JToggleButton, для которых назначены иконки |
| 6 | JDesktopPane | Панель — рабочий стол. Еще один тип панели с абсолютным позиционированием элементов |
| **AWT (управляющие элементы пакета AWT)** | | |
| 1…13 | Label, Button, TextField, TextArea, CheckBox, Choice, List, ScrollBar, ScrollPane, Panel, Canvas, MenuBar, PopupMenu | Устаревший набор компонентов. Может представлять интерес только для тех разработчиков, кому необходимо писать прило-жения Java, работающие с самыми древними версиями Java-машин |
| **Beans** (компоненты, установленные пользователем) | | |
| 1 | Coose Bean | Выберите компонент. Это не компонент, а диалог добавления компонента в палитру на страницу Beans. О том, как добавить компонент на нужную страницу палитры, рассказывается в *главе 15* |
| **Java Persistence** (компоненты технологии Java Persistence) | | |
| 1 | EntityManager | Менеджер сущностей. Используется в технологии Java Persistence для управления сущностями (entities) — объектами, которые могут сохраняться после окончания работы приложения и загружаться "из внешнего мира" (например, из базы данных) |
| 2 | Query | Запрос в технологии Java Persistence |
| 3 | Query Result | Результат запроса в технологии Java Persistence (имеет тип List) |

**Булевый (логический) тип**

boolean a;

boolean b;

Таблица 3.1. Логические операторы

|  |  |  |
| --- | --- | --- |
| **Оператор** | **Название** | **Пример** |
| **&&** | and (логическое И) | a&&b |
| **||** | or (логическое ИЛИ) | a||b |
| **^** | xor (логическое исключающее ИЛИ) | a^b |
| **!** | not (логическое НЕ) | !a |

Таблица 3.2. Таблица истинности для булевых операторов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Выражение** | | | **Значения** | | |
| **a** | 0 | 0 | | 1 | 1 |
| **b** | 0 | 1 | | 0 | 1 |
| **a&&b** | 0 | 0 | | 0 | 1 |
| **a||b** | 0 | 1 | | 1 | 1 |
| **a^b** | 0 | 1 | | 1 | 0 |
| **!a** | 1 | 1 | | 0 | 0 |

***Таблица 3.3.*** *Операторы сравнения и принадлежности*

|  |  |  |
| --- | --- | --- |
| **Оператор** | **Название** | **Пример** |
| **==** | Равно | a==b |
| **!=** | Не равно | a!=b |
| **>** | Больше | a>b |
| **<** | Меньше | a<b |
| **>=** | Больше или равно | a>=b |
| **<=** | Меньше или не равно | a<=b |
| **instanceof** | Принадлежность объекта классу *(см. разд. 6.12 и 9.3)* | obj instanceof MyClass |

**Таблица 3.4. Целочисленные типы**

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип** | **Число байтов** | **Диапазон значений** | **Описание** |
| **byte** | 1 | -128..127 | Однобайтовое целое число (8-битовое целое со знаком) |
| **short** | 2 | -215..215-1 =  -32768.. 32767 | Короткое целое число (16-битовое целое со знаком) |
| **char** | 2 | \u0000..\uFFFF=0.. 65535 | Символьный тип (беззнаковое 16-битовое целое) |
| **int** | 4 | -231..231-1 =  -2.147483648·109..  2.147483647·109 | Целое число (32-битовое целое со знаком) |
| **long** | 8 | -263..263-1 =  -9.22337203685478·1018..  9.22337203685478·1018 | Длинное целое число (64-битовое целое со знаком) |

**int i,j,k;**

**int j1;**

**byte i1,i2=-5;**

**short i3=-15600;**

**long m1=1,m2,m3=-100;**

Таблица 3.5. Основные операторы для работы с целочисленными величинами

|  |  |  |  |
| --- | --- | --- | --- |
| **Оператор** | **Название** | **Пример** | **Примечание** |
| **+** | Оператор сложения | i+j | Если операнды i и j имеют разные типы или типы byte, short или char, то действуют правила автоматического преобразования типов |
| **–** | Оператор вычитания | i-j | |
| **\*** | Оператор умножения | i\*j | |
| **/** | Оператор деления | i/j | Результат округляется до целого путем от-брасывания дробной части как для положи-тельных, так и для отрицательных чисел |
| **%** | Оператор остатка от целочисленного деления | i%j | Возвращается остаток от целочисленного деления |
| **=** | Оператор присваива-ния | v=i | Сначала вычисляется выражение i, после чего полученный результат копируется в ячейку v |
| **++** | Оператор инкремента (увеличения на 1) | v++  ++v | v++ эквивалентно v=v+1  ++v эквивалентно v=v+1  О различии префиксной и постфиксной форм *см. разд. 5.5* |
| **--** | Оператор декремента (уменьшения на 1) | v--  --v | v-- эквивалентно v=v-1  --v эквивалентно v=v-1  О различии префиксной и постфиксной форм *см. разд. 5.5* |
| **+=** | v+=i | v+=i эквивалентно v=v+i | |
| **-=** | v-=i | v-=i эквивалентно v=v-i | |
| **\*=** | v\*=i | v\*=i эквивалентно v=v\*i | |
| **/=** | v/=i | v/=i эквивалентно v=v/i | |
| **%=** | v%=i | v%=i эквивалентно v=v%i | |

Таблица 3.6. Вещественные типы

|  |  |  |  |
| --- | --- | --- | --- |
| **float** | 4 байта  (32 бита) | 1.5·10-45..3.4·1038 | Одинарная точность, 7—8 значащих десятичных цифр мантиссы.  Тип real\*4 стандарта IEEE754 |
| **double** | 8 байт  (64 бита) | 5·10-324..1.7·10308 | Двойная точность, 15—16 значащих цифр мантиссы.  Тип real\*8 стандарта IEEE754 |

***Таблица 3.7.*** *Основные операторы для работы с вещественными типами*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Оператор** | **Название** | **Пример** | **Примечание** | |
| **+** | Оператор сложения | x+y | В случае, когда операнды x и y имеют раз-ные типы, действуют правила автоматиче-ского преобразования типов | |
| **–** | | Оператор вычитания | | x-y |
| **\*** | | Оператор умножения | | x\*y |
| **/** | Оператор деления | x/y | Результат является вещественным. В слу-чае, когда операнды x и y имеют разные типы, действуют правила автоматического преобразования типов | |
| **%** | Оператор остатка от целочисленного деления | x%y | Возвращается остаток от целочисленного деления x на y. В случае, когда операнды x и y имеют разные типы, действуют правила автоматического преобразования типов | |
| **=** | Оператор присваивания | v=x | Сначала вычисляется выражение x, после чего полученный результат копируется в ячейку v | |
| **++** | Оператор инкремента (увеличения на 1) | v++  ++v | Эквивалентно v=v+1 | |
| **--** | Оператор декремента (уменьшения на 1) | v--  --v | Эквивалентно v=v-1 | |
| ***+=*** | | v+=x | | Эквивалентно v=v+x |
| ***-=*** | | v-=x | | Эквивалентно v=v-x |
| ***\*=*** | | v\*=x | | Эквивалентно v=v\*x |
| ***/=*** | | v/=x | | Эквивалентно v=v/x |
| ***%=*** | | v%=x | | Эквивалентно v=v%x |

**Таблица 3.8. Функции, заданные в классе Math**

|  |  |
| --- | --- |
| **Функция** | **Примечание** |
| **toRadians(angdeg)** | angdeg / 180.0 PI; — перевод углов из градусов в радианы |
| **toDegrees(angrad)** | angrad 180.0 / *PI*; — перевод углов из радиан в градусы |
| **Степени, экспоненты, логарифмы** | |
| **exp(x)** | *ex — экспонента* |
| **expm1(x)** | *ex* – 1. При *x*, близком к 0, дает гораздо более точные значения, чем exp(*x*) –1 |
| **log(x)** | ln(*x*) — натуральный логарифм |
| **log10(x)** | log10(*x*) — десятичный логарифм |
| **log1p(x)** | ln(1 + *x*). При *x*, близком к 0, дает гораздо более точные значения, чем log(1 + *x*) |
| **sqrt(x)** | — квадратный корень x |
| **cbrt(x)** | — кубический корень 3 x |
| **hypot(x,y)** | — вычисление длины гипотенузы по двум катетам 2 2 x y |
| **pow(x, y)** | *xy* — возведение *x* в степень *y* |
| **sinh(x)** | sh(x)= — гиперболический синус 2 x x e e |
| **cosh(x)** | ch(x)= — гиперболический косинус 2 x x e e |
| **tanh(x)** | th(x)= — гиперболический тангенс x x x x e e e e |
| **Модуль, знак, минимальное, максимальное число** | |
| **abs(m), abs(x)** | Абсолютное значение числа. Аргумент типа int, long, float или double. Результат того же типа, что аргумент |
| **signum(a), signum(x)** | Знак числа. Аргумент типа float или double. Результат того же типа, что аргумент |
| **min(m,n), min(x,y)** | Минимальное из двух чисел. Аргументы одного типа. Возможны типы: int, long, float, double. Результат того же типа, что аргумент |
| **max(m,n), max(x,y)** | Максимальное из двух чисел. Аргументы одного типа. Возможны типы: int, long, float, double. Результат того же типа, что аргумент |
| **Округления** | |
| **ceil(x)** | Ближайшее к *x* целое, большее или равное *x* |
| **floor(x)** | Ближайшее к *x* целое, меньшее или равное *x* |
| **round(a)**  **round(x)** | Ближайшее к *x* целое. Аргумент типа float или double. Результат типа long, если аргумент double, и типа int — если float. То же, что (int)floor(x + 0.5) |
| **rint(x)** | Ближайшее к *x* целое |
| **ulp(a), ulp(x)** | Расстояние до ближайшего большего чем аргумент значения того же типа (дискретность изменения чисел в формате с плавающей точкой вблизи данного значения). Аргумент типа float или double. Результат того же типа, что аргумент |
| **Случайное число, остаток** | |
| **random()** | Псевдослучайное число в диапазоне от 0,0 до 1,0. При этом: 0 <= Math.random() < 1 |
| **IEEEremainder(x,y)** | Погрешность целочисленного деления *x* на *y*, т. е. разность: *x* — *y* \* *n*, где *n* — результат целочисленного деления |

**Таблица 3.9. Сопоставление примитивных типов и оболочечных классов**

|  |  |
| --- | --- |
| **Примитивный тип** | **Оболочечный класс** |
| **byte** | **Byte** |
| **short** | **Short** |
| **char** | **Character** |
| **int** | **Integer** |
| **long** | **Long** |
| **float** | **Float** |
| **double** | **Double** |

**Таблица 3.10. Таблица приоритета операторов**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Приоритет | Группа операторов | Операторы | | | | |
| 1 выс ший | Постфиксные | ( ) | [ ] | . | | |
| 2 | Унарные | ++ опер  опер ++ | ––опер  опер–– | ~ | ! | + опер  – опер |
| 3 | Создания объектов и преобразования типа | new | (тип)  операнд | | | |
| 4 | Мультипликативные | \* | / | % | | |
| 5 | Аддитивные | + | – | | | |
| 6 | Сдвиги битов | >> | >>> | << | | |
| 7 | Отношения | > | >= | < | <= | instanceof |
| 8 | Эквивалентности | == | != | | | |
| 9 | Побитовое И | & | | | | |
| 10 | Побитовое исклю-чающее ИЛИ | ^ | | | | |
| 11 | Побитовое ИЛИ | | | | | | |
| 12 | Логическое И | && | | | | |
| 13 | Логическое ИЛИ | || | | | | |
| 14 | Условный | ? : | | | | |
| 15 низш | Присваивания | = | Оператор = ( +=, –=, \*=, /= и т. п.) | | | |

**Примеры допустимых присваиваний переменным числовых типов**

byte byte0 = 1; //-128..127

short short0 = 1;//-32768.. 32767

char char0 = 1;//0.. 65535

int int0 = 1; //-2.147483648E9.. 2.147483647E9

long long0 = 1;//-9.223E18.. 9.223E18

float float0 = 1;// ±(1.4E-45..3.402E38)

double double0 = 1;// ±(4.9E-324..1.797E308 )

short0 = byte0;

byte0 = (byte)short0;

char0 = (char)short0;

int0 = short0;

int0 = char0;

char0 = (char)int0;

short0 = (short)int0;

long0 = byte0;

byte0 = (byte)long0;

long0 = char0;

long0 = int0;

int0 = (int)long0;

float0 = byte0;

float0 = int0;

float0 = (float)double0;

double0 = float0;

**Упаковка и распаковка при присваиваниях и в выражениях**

Integer obj1 = 10; //упаковка

int i1 = obj1\*2; //распаковка при вычислении

выражения

Byte b = 1; //упаковка

obj1 = i1/10; //упаковка

b = 2; //упаковка

**оболочечных классах хранятся константы**

Byte.MIN\_VALUE

Byte.MAX\_VALUE

Float.MIN\_VALUE

Float.MAX\_VALUE

Double.MIN\_VALUE

Double.MAX\_VALUE

оболочечных классах также имеются методы классов

Byte.parseByte(*строка*)

Short.parseShort(*строка*)

Integer.parseInt(*строка*)

Long.parseLong(*строка*)

Float.parseFloat(*строка*)

Double.parseDouble(*строка*)

Byte.valueOf(*строка*)

Short.valueOf(*строка*)

Integer.valueOf(*строка*)

Long.valueOf (*строка*)

Float.valueOf (*строка*)

Double.valueOf (*строка*)

int n1=Integer.MAX\_VALUE;

double d1= Double.MIN\_VALUE;

Отметим, что присваивание

double d2= Double.parseDouble(jTextField1.getText());

функционально будет работать совершенно так же, как:

double d2= Double.valueOf(jTextField1.getText());

несмотря на то, что во втором случае методом valueOf создается объект оболочечного типа Double.

**Правила явного и автоматического преобразования типа при работе с числовыми величинами**

Виртуальный процессор Java-машины может проводить на аппаратном уровне цело-численные математические вычисления только с величинами типа int или long. При этом операнды в основных математических операциях (+, –, \*, /, %) должны быть одного типа.

В тех случаях, когда операнды имеют разные типы или типы byte, short или char, дейст-вуют правила автоматического преобразования типов: для величин типа byte, short или char сначала происходит преобразование в тип int, после чего производится их подста-новка в качестве операндов. Если же один из операндов имеет тип long, то действия производятся с числами типа long, поскольку второй операнд автоматически преобра-зуется к этому типу.

Аналогично при работе с вещественными величинами в интерпретаторе Java возможна работа на аппаратном уровне (математический сопроцессор) только с операндами ти-пов float и double. При этом операнды в основных математических операциях должны быть одного типа. Если один из операндов имеет тип double, а другой float, то действия производятся с числами типа double, поскольку операнд типа float автоматически пре-образуется к типу double.

Если один из операндов целочисленный, а другой вещественный, то сначала идет пре-образование целочисленного операнда к такому же вещественному типу, а уже потом выполняется оператор.

Рассмотрим теперь правила совместимости типов по присваиванию. Они просты: диа-пазон значений типа левой части не должен быть у´же, чем диапазон типа правой. По-этому в присваиваниях, где тип в правой части не умещается в диапазон левой части, требуется указывать *явное преобразование типа*. Иначе компилятор выдаст сообщение об ошибке с не очень адекватной диагностикой:

**possible loss of precision (возможная потеря точности).**

Для явного преобразования типа в круглых скобках перед преобразуемой величиной ставят имя того типа, к которому нужно преобразовать.

(float)d

(double)f

Во многих случаях к явному преобразованию типов прибегать не нужно, т. к. действует автоматическое преобразование, если переменной, имеющей тип с более широким диа-пазоном изменения, присваивается выражение, имеющее тип с более узким диапазоном изменения.

Таблица 4.2. Побитовые операторы

|  |  |  |
| --- | --- | --- |
| **Оператор** | **Название** | **Пример** |
| ~ | Оператор побитового дополнения (побитовое НЕ, побитовое отрицание) | ~i |
| ^ | Оператор XOR (побитовое исключающее ИЛИ) | i^j |
| & | Оператор AND (побитовое И) | i&j |
| | | Оператор OR (побитовое ИЛИ) | i|j |
| << | Оператор левого побитового сдвига | m<<n |
| >>> | Оператор беззнакового правого побитового сдвига | m>>>n |
| >> | Оператор правого побитового сдвига с сохранением знака отриц-го числа | m>>n |
| &= | y&=x эквивалентно y=y&x | m&=n |
| |= | y|=x эквивалентно y=y|x | m|=n |
| ^= | y^=x эквивалентно y=y^x | m^=n |
| >>= | y>>=x эквивалентно y= y>>x | m>>=n |
| >>>= | y>>>=x эквивалентно y= y>>>x | m>>>=n |
| <<= | y<<=x эквивалентно y= y<<x | m<<=n |

Таблица 4.3. Операция OR с битовой маской

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **a** | a3 | a2 | a1 | a0 |
| **m** | 0 | 1 | 0 | 1 |
| **a|m** | a3 | 1 | a1 | 1 |

Таблица 4.4. Операция AND с битовой маской

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **a** | a3 | a2 | a1 | a0 |
| **m** | 0 | 1 | 0 | 1 |
| **a&m** | 0 | a2 | 0 | a0 |

Таблица 4.5. Операция XOR с битовой маской

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **a** | 1 | 1 | 0 | 0 |
| **m** | 0 | 1 | 0 | 1 |
| **a^m** | 1 | 0 | 0 | 1 |

Таблица 4.6. Повторная операция XOR с битовой маской

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **a^m** | 1 | 0 | 0 | 1 |
| **m** | 0 | 1 | 0 | 1 |
| **(a^m)^m** | 1 | 1 | 0 | 0 |