Валерий Яценков

Java за неделю

Основы работы с NetBeans IDE 8.2

Графический оконный интерфейс

Примеры и полезные советы

Лямбда-выражения

Многопоточность

2018

Валерий Яценков

Java за неделю. Вводный курс

«Издательские решения»

Яценков В. С.

Java за неделю. Вводный курс / В. С. Яценков — «Издательские решения»,

ISBN 978-5-44-904684-0

Вводный курс для начинающих изучать язык Java. Быстрый и ощутимый результат — лучший стимул в обучении. Занимаясь 1—2 часа по вечерам, уже через неделю вы будете писать полноценные оконные приложения с графическим интерфейсом. Книга содержит примеры с пошаговыми пояснениями. Приведен список литературы для желающих продолжить обучение. Работа с учебными проектами полностью ведется в современной визуальной среде разработки NetBeans IDE. Книгу дополняет файловый архив с исходными кодами примеров.

ISBN 978-5-44-904684-0

Содержание

Java за неделю
Вводный курс

Валерий Станиславович Яценков

ISBN 978-5-4490-4684-0
Создано в интеллектуальной издательской системе Ridero

Часть I. Теория

Глава 1. Введение

Язык программирования – это инструмент решения прикладных задач. В идеале разработчик должен хорошо разбираться в нескольких языках программирования и подбирать инструмент под задачу, а не пытаться подогнать задачу под возможности инструмента.

После прочтения этой книги вы получите достаточно полное представление о языке Java и его возможностях. Может даже оказаться, что он не подходит для ваших *сегодняшних* задач. Замечательно! – вы не потеряете напрасно время на чтение толстых учебников и углубленное изучение языка. Зато вы будете хорошо знать, для чего пригодится язык Java, и сможете вернуться к нему в любое время. Если Java – именно то, что вам сейчас нужно, то после прочтения этой книги будет легче приступить к углубленному изучению языка.

Объем и сложность материала вводного курса подобраны таким образом, чтобы уделяя по вечерам 1—2 часа на чтение и работу с компьютером, вы приблизительно за неделю смогли овладеть навыками программирования на языке Java в среде разработки NetBeans.

Разумеется, эта книга станет лишь первым шагом в изучении языка Java и среды разработки NetBeans. Впереди вас ждет поиск и усвоение огромного объема информации.

1.1 Особенности текста книги и архив файлов

Книга подготовлена и опубликована при помощи издательского сервиса Ridero. Это новый проект, который помогает издавать книги быстрее и делать их дешевле и доступнее.

Но технические возможности издателя пока не полностью адаптированы к изданию технических текстов. Например, система набора текста автоматически заменяет в листингах двойные «технические» кавычки на «лингвистические», двойной минус (декремент) на длинное тире. Мы просим отнестись с пониманием к этим мелким временным недостаткам издательского сервиса.

В файловом архиве книги вы найдете полные исходные коды всех примеров программ из книги, а также дополнительные файлы с наборами иконок для графического интерфейса. Архив можно скачать из файловых хранилищ

Dropbox:
https://www.dropbox.com/s/wo0u8916cnyc31p/Java_Files.zip?dl=0

Яндекс Диск:
https://yadi.sk/d/fIoAfXyp3Sj8gP

1.2 Идеология Java

Разработка языка Java началась в 1990 году под названием Oak (дуб) – не самое лучшее название для интеллектуального продукта. В процессе работы значительно изменилась концепция языка, а затем и его название. Окончательный вариант открытого и общедоступного языка Java был обнародован в 1995 году.

Нельзя сказать, что новый язык легко и быстро завоевал популярность, но сегодня это самый востребованный язык программирования. Он удачно занял нишу языка для приложений массового пользования, широко распространяемых через Интернет. Для таких приложе-

ний важна *независимость от платформы* – работа прикладной программы не должна зависеть от аппаратной части компьютера и операционной системы.

При распространении программ на языке Java не возникает проблем с отсутствием на компьютере пользователя нужных программных библиотек или модулей. Дистрибутив программы на языке Java, как правило, состоит из одного файла, который содержит в себе всё необходимое для работы приложения на любом компьютере с установленной Java-машиной. Впрочем, в состав дистрибутива иногда могут входить отдельные внешние файлы настроек или базы данных, которые невозможно упаковать внутрь файла скомпилированного приложения.

Язык Java популярен еще и потому, что применяется при разработке приложений Android. Можно писать приложения на «чистом» языке Java (в реализации Java Mobile) или использовать среду разработки, предоставляющую расширенные возможности. В любом случае знание Java является обязательным условием для разработчика приложений Android.

Официальная среда разработки программ на Java полностью бесплатная, включая большое количество дополнительных модулей и библиотек, разработанных сообществом программистов. Существуют платные инструменты разработки, но мы прекрасно обойдемся без них.

У языка Java низкий порог вхождения – первые полезные приложения с полноценным графическим интерфейсом можно создавать через несколько дней после начала изучения языка. По этой причине язык Java очень популярен, например, среди радиолюбителей, которые разрабатывают собственные приложения для взаимодействия компьютеров с электронными устройствами.

Разумеется, Java широко применяется в профессиональной среде. Это мощный язык программирования с поддержкой многопоточности, на котором разработано большое количество коммерческих приложений.

Давайте разберемся, как работает Java—программа.

1.3 Как работает Java

Языки программирования общего назначения можно разделить на *интерпретирующие* и *компилирующие*.

В первом случае специальная программа—интерпретатор поочередно преобразовывает каждую строку программы в команды процессора и отправляет их на выполнение под управлением операционной системы. Поэтому для каждого типа процессора и операционной системы нужна отдельная версия интерпретатора. Интерпретируемые программы работают медленнее, чем скомпилированные, потому что построчное преобразование программы в двоичный код занимает больше времени, чем выполнение готового кода. Но существуют ситуации, когда применение интерпретатора оправдано.

Во втором случае компилятор заранее и полностью преобразует программу в бинарный процессорный код. Эта процедура выполняется один раз. Далее программа распространяется в виде готового кода и может быть запущена без участия компилятора. При этом тоже необходимо обеспечить совместимость кода программы с процессором и операционной системой компьютера пользователя.

Java – необычный язык программирования. При компиляции программа на языке Java превращается в специальный *байт-код*. Он представляет собой набор унифицированных инструкций для специальной *Java-машины (Java Virtual Machine, JVM)*, установленной на компьютере. Иными словами, программа выполняется внутри виртуальной машины, которая служит «посредником» между программой и компьютером.

Совместимость вашего приложения с аппаратной частью компьютера и операционной системой обеспечивается виртуальной машиной. Вы можете из программы обращаться к пор-

там компьютера, файловой или графической системе, и совершенно не задумываться о том, как это будет реализовано. Об этом позаботились разработчики нужной версии Java-машины.

Таким образом, вместо того, чтобы разрабатывать разные версии прикладной программы, достаточно установить на компьютер готовую и бесплатную Java-машину, которая учитывает и реализует особенности операционной системы. Виртуальные машины Java для большинства операционных систем можно скачать на сайте **www.java.com**. В операционную систему Android поддержка Java встроена по умолчанию.

Java-машина не занимает много места в памяти компьютера. Времена, когда программы на языке Java долго запускались и медленно работали, остались в прошлом. Сейчас они лишь незначительно отстают в быстродействии от обычных скомпилированных программ.

Чтобы избежать путаницы, отметим, что язык JavaScript не имеет ничего общего с языком Java. Это язык для написания сценариев (скриптов), которые включены в состав HTML-страниц и выполняются средствами браузера. Слово «Java» было добавлено компанией Netscape – разработчиком языка JavaScript – исключительно из маркетинговых соображений.

1.4 Что читать дальше?

О программировании на языке Java издано много хороших книг, в том числе на русском языке. Настоятельно рекомендую несколько изданий, которые особенно хороши для знакомства с Java:

Хабибуллин И. Ш. Самоучитель Java. – 3-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2008. – 768 с.

Хабибуллин И. Ш. Java 7 в подлиннике. – СПб.: БХВ-Петербург, 2012. – 768с.

Прохоренок Н. А. Основы Java. – СПб.: БХВ-Петербург, 2017. – 704 с.

Васильев А. Н. Программирование на Java для начинающих. – Москва: Издательство «Э», 2017. – 704с.

Монахов В. В. Язык программирования Java и среда NetBeans. – СПб.: БХВ-Петербург, 2012. – 704с. + DVD.

1.5 Другие книги автора

Друзья, если вы интересуетесь техническим творчеством и программированием микроконтроллеров, вам могут пригодиться эти книги:

https://www.ozon.ru/context/detail/id/141872715/

https://www.ozon.ru/context/detail/id/135412298/

Глава 2. Подготовка к работе с Java

Давно остались в прошлом времена, когда программист набирал исходный код программы в текстовом редакторе, а затем запускал компилятор в командной строке и мучительно пытался разобраться в сообщениях об ошибках. Теперь любой серьезный язык программирования располагает *интегрированной средой разработки (Integrated Development Environment, IDE)*. Это специальный набор инструментов разработчика, который может включать в себя редактор со множеством удобных функций, средство управления проектами, компилятор, отладчик, эмулятор мобильных устройств, справочную систему и многое другое.

В настоящее время для программирования на языке Java применяется несколько популярных сред разработки: NetBeans, Eclipse, JDeveloper, JBuilder, IntelliJ IDEA.

В этой книге вы познакомитесь с бесплатной средой NetBeans, которая разработана корпорацией Sun и распространяется с открытым исходным кодом. Допускается использование среды в коммерческих разработках. Название среды содержит игру слов: Java – сорт кофе, Beans – зерна. В тоже время словосочетание Net Beans можно перевести как «сетевые компоненты», потому что дополнительные компоненты IDE можно скачивать из сети по мере необходимости.

Пусть вас не смущает бесплатность и открытость исходного кода NetBeans IDE. По мнению многих профессиональных программистов, эта среда является самой удобной и развитой. Она оснащена отличным редактором графических интерфейсов и продвинутым средством разработки приложений для мобильных устройств.

Возможности среды можно расширять при помощи бесплатных плагинов, которые размещены в специальном репозитории.

2.1 Устанавливаем JDK и NetBeans

Чтобы приступить к программированию на Java, вы должны установить на свой компьютер два обязательных компонента: JDK и NetBeans.

JDK (Java Development Kit) – средство разработки, в состав которого входит компилятор, библиотеки, справочная документация, и собственно сама среда выполнения программ JRE (Java Runtime Environment). В принципе, можно обойтись этим набором, но вам придется набирать код программы в каком-то текстовом редакторе и вручную запускать компилятор из командной строки. В таком случае не может быть речи о средствах отладки или визуального редактирования интерфейсов.

Для полноценной и комфортной работы после установки JDK необходимо установить оболочку NetBeans. Причем установку следует выполнять именно в таком порядке – сначала JDK, затем NetBeans. В противном случае вам придется вручную указать пути к файлам JDK в настройках NetBeans. Но мы поступим еще проще и воспользуемся составным пакетом «JDK + NetBeans Bundle». Установщик пакета сделает за нас всю работу.

Войдите на сайт Oracle по адресу

http://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-142931.html

Если к тому моменту, когда вы читаете эту книгу, изменится версия JDK или NetBeans IDE, то указанная ссылка может перестать работать. В таком случае введите в любую поисковую систему ключевые слова «JDK NetBeans IDE bundle», и это будет проще, чем искать новую ссылку на запутанном корпоративном сайте Oracle.

Щелкните по пункту «Accept License Agreement» (Принять лицензионное соглашение) и скачайте установочный файл для своей операционной системы. Обратите внимание на соответствие разрядности. В примере на рис. 2.1 выбран пакет для 32—разрядной ОС Windows.

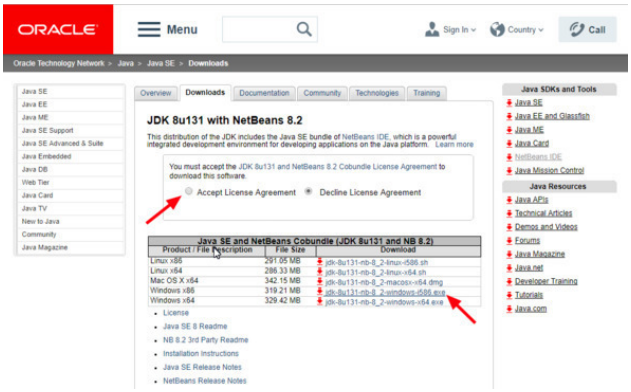

Рис. 2.1 Выбор установочного файла JDK + NetBeans bundle

Запустите установочный файл и согласитесь со всеми пунктами настроек. На момент подготовки книги распространялась версия NetBeans IDE 8.2. При первом запуске NetBeans наверняка обнаружит обновления и предложит загрузить их. После установки обновлений можно приступать к знакомству с интерфейсом NetBeans и написанию первой программы.

2.2 Соглашение об именах

Прежде, чем приступить к созданию первого проекта, сделаем небольшое отступление и перечислим основные правила составления имен в языке Java. Обязательно ли нужно выполнять эти правила? Если на званом ужине вы будете вытирать руки о скатерть, то вас не прогонят из-за стола. Но второй раз не пригласят. С выполнением общепринятых соглашений в программировании похожая ситуация. Отклонение от правил именования в большинстве случаев не вызовет ошибку компиляции, но затруднит понимание исходного кода другими программистами. Более того, даже вам будет трудно разрабатывать и отлаживать собственный код, обращаясь к чужим примерам с корректно заданными именами. Надо с первых шагов приучить себя к строгому соблюдению как формальных, так и неписаных правил программирования. Работоспособность приложения – не единственный критерий качества кода.

> Язык Java регистрозависимый. Например, filesize и fileSize – разные имена. Тем не менее, лучше избегать использования имен, которые различаются лишь регистром символов, чтобы не затруднять понимание и отладку программы. Обычно «горбатый регистр» (Camel casing) применяется для выделения первых букв слова в составном имени, например, MyFirstClass.

Итак, вот пункты соглашения об именах Java (в скобках приведены примеры):

Пакеты и подпакеты – существительные в единственном числе, только в нижнем регистре, в составных именах слова разделяются подчеркиванием (input_control).

Классы и интерфейсы – существительные или словосочетания в значении существительного. Первые буквы слов в верхнем регистре, слова не разделяются (UserInfo). Имена классов—исключений заканчиваются словом Exception (InvalidCountException).

Классы—наследники – рекомендуется использовать имена, в которых содержится имя родительского класса (LocalConnect extends Connect). Исключение составляют имена классов—наследников, из которых очевидно, что они наследуют суперкласс (Oval extends Figures).

Поля и локальные переменные – существительные в нижнем регистре (size). Если название составное, то следующие слова начинаются с заглавной буквы, разделители не используются (imageHeight). Имена переменных должны соответствовать типу хранимых данных. Например, имя переменной currentUser интуитивно соответствует номеру пользователя (целое

число). Для хранения имени пользователя (строка) лучше использовать переменную с именем currentUserName.

Переменные типа static final – существительные или словосочетания в верхнем регистре, слова разделены подчеркиваниями (MAIN_COLOUR).

Методы – глаголы в нижнем регистре (calculate) или словосочетания, отражающие действие (printAmount). Глаголы должны максимально полно и точно описывать действие, которое выполняет метод.

Имена методов, выполняющих чтение или изменение значений полей класса, должны начинаться на get и set соответственно (getFileSize, setFontColour). Исключение составляют методы, возвращающие значения полей типа boolean. Их имена должны начинаться на is (isFileOpen).

Имена методов, выполняющих преобразование к другому типу данных, начинаются на to (toString).

Имена методов, которые создают и возвращают объект, начинаются с create (createDataset).

Имена методов, инициализирующих поля класса или элементы графического интерфейса, начинаются с init (initWindow) и применяются только в конструкторе класса.

2.2.1 Зарезервированные слова и литералы

В таблице 2.1 приведены ключевые слова, зарезервированные для синтаксических конструкций языка Java. Они не могут быть именами переменных, классов и т.п., их нельзя переопределять.

Таблица 2.1 Зарезервированные слова языка Java

abstract	assert	boolean	break	byte
case	catch	char	class	const
continue	default	do	double	else
enum	extends	final	finally	float
for	goto	if	implements	import
instanceof	int	interface	long	native
new	package	private	protected	public
return	short	static	strictfp	super
switch	synchronized	this	throw	throws
transient	try	void	volatile	while

2.3 Первый проект на Java

Сейчас вы создадите свой первый проект. Возможно, у вас появится много вопросов, но не волнуйтесь – ответы будут даны позже. Сначала вы должны получить базовые навыки работы с NetBeans IDE, чтобы использовать примеры по мере прочтения книги.

Запустите NetBeans. Выберите пункты меню **Файл | Создать проект** или нажмите на значок с изображением зеленой папки и символа «+». Выберите категорию **Java** и тип проекта **Приложение Java** (рис. 2.2). Введите имя нового проекта. Пусть это будет HelloJava (рис. 2.3).

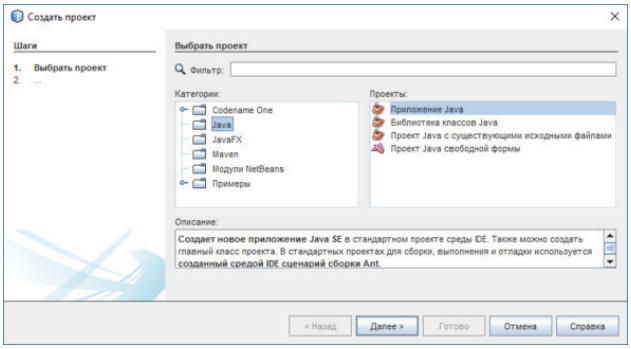

Рис. 2.2 Выберите тип проекта Java

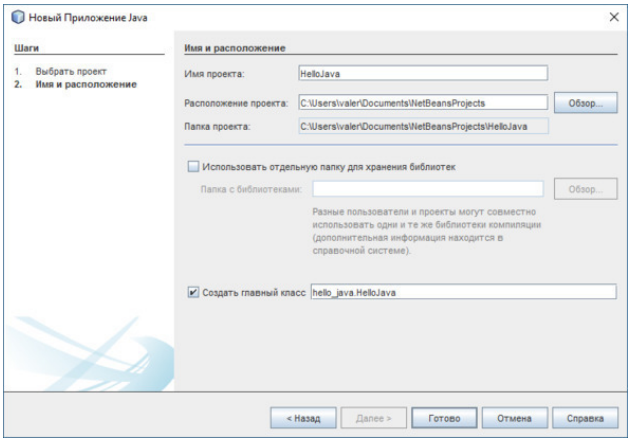

Рис. 2.3 Введите название проекта

Обратите внимание, что вам предложено создать *главный класс*, для которого автоматически сформировано имя *пакета*. Так как название пакета состоит из двух слов, поставьте между ними подчеркивание, чтобы имя пакета полностью соответствовало соглашению об именах. Нажмите кнопку **Готово**, и через несколько секунд NetBeans сформирует новый проект и откроет шаблон исходного кода.

Пока не обращайте внимания на серые строки комментариев, где предложено ввести информацию о лицензии и авторе проекта. Если эти комментарии мешают, удалите их.

Найдите строку

```
// TODO code application logic here
```

Она указывает на место, в которое необходимо вставить основной исполняемый код. Вместо этой строки введите

```
System.out.println («Hello Java»);
```

У вас должна получиться программа как в листинге 2.1 (комментарии удалены).

Листинг 2.1 Первая программа на языке Java

```
package hello_java;

public class HelloJava {

public static void main (String [] args) {
System.out.println («Hello Java»);
}
```

```
}
```

Запустите программу на выполнение, нажав значок зеленого треугольника или выбрав пункт меню **Выполнить | Запустить проект**. Спустя несколько секунд сборка проекта будет завершена. В нижней части интерфейса NetBeans откроется окно терминала, в который будет выведен текст «Hello Java» и сообщение об успешной сборке проекта (рис. 2.4).

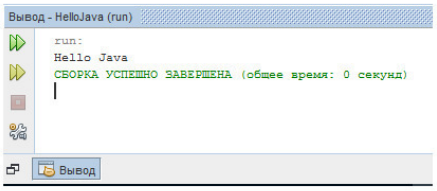

Рис. 2.4 Окно системного терминала NetBeans

Попробуйте совершить ошибку в тексте программы и посмотрите, как отреагирует среда разработки. Удалите одну из кавычек, обрамляющих строку «Hello Java». Система контроля синтаксиса немедленно отреагирует на ошибку. Ближайшая круглая скобка будет выделена красным цветом (из-за отсутствующей кавычки эта скобка оказалась не на своем месте), а напротив строки, содержащей ошибку, появился восклицательный знак на красном фоне. Это обозначение критической ошибки, которая приведет к ошибке компиляции. При наведении указателя мыши на значок ошибки появляется всплывающая подсказка (рис. 2.5).

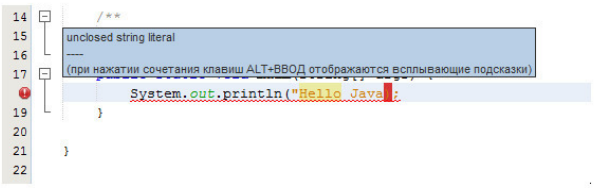

Рис. 2.5 Система проверки синтаксиса в действии

Теперь сделайте ошибку в названии пакета, и вместо hellojava в первой строке введите yellojava. Слева от строки вновь появился значок, только теперь это лампочка с маленьким восклицательным знаком. Это означает, что система не видит здесь фатальную синтаксическую ошибку, которая требует обязательной правки кода, а лишь уведомляет, что вы что-то перепутали или упустили. В данном случае вы ссылаетесь на пакет, которого нет в проекте. Если вы и в самом деле включите в состав проекта пакет с названием yellojava, то значок ошибки исчезнет.

Если вопреки сообщениям об ошибке принудительно запустить компиляцию проекта, то в окне системного терминала будет выведено диагностическое сообщение с указанием строки (или нескольких строк), где присутствуют ошибки. Щелкните на ссылку в сообщении, и курсор в окне редактора автоматически переместится на нужную строку программы.

2.4 Забегая вперед: классы, объекты и методы

Изучение сложного языка программирования – это борьба за первенство между курицей и яйцом. Чтобы понять программу на языке Java, необходимо владеть основными понятиями объектно-ориентированного программирования (ООП). С другой стороны, чтобы изучить понятия ООП применительно к Java, сначала надо познакомиться с синтаксисом и операторами. Если вы уже знакомы с ООП по другим языкам, то вам будет намного проще изучать Java.

Чтобы продолжить рассказ о языке Java и среде разработки, я немного забегу вперед и скажу несколько слов о классах, объектах и методах. Более подробно об этом будет рассказано в главе 6 «Классы и объекты». Если есть желание, можете перейти к чтению главы 6 прямо сейчас, а затем вернуться к главе 2.

Итак, любая программа Java состоит из *классов*, на основе которых создаются *объекты*. Объект в общем случае представляет собой набор переменных и *методов*. Метод – это именованный фрагмент кода, предназначенного для обработки переменных объекта и выполнения иных действий.

Программа практически всегда содержит главный класс и главный метод main (), который выполняется при запуске программы.

Вернемся к листингу 2.1. В нем объявлен главный класс HelloJava, который содержит единственный метод main (). Если вы не объявили главный класс при создании нового проекта, то впоследствии компилятор все равно спросит вас, какой класс считать главным.

2.5 Структура проекта Java

Современные программы давно перестали состоять из одного файла, поэтому теперь вместо «программа» принято говорить «проект». Как вы сейчас увидите, даже если имеется всего один файл исходного кода, проект приложения на языке Java включает в себя и другие компоненты. Далее в книге мы будем применять термин «программа» только к ограниченным фрагментам кода в примерах или к отдельным файлам кода. Говоря о приложении в целом, будем использовать слов «проект».

На вершине иерархии Java располагается собственно проект, с создания которого мы начинаем свою работу (рис. 2.6). В нашем случае это проект под названием HelloJava. Проект состоит из одного или нескольких пакетов исходных кодов, а также подключаемых библиотек.

Даже небольшой учебный проект может состоять из нескольких десятков классов. Серьезные коммерческие проекты, которые разработаны коллективом программистов, состоят из тысяч классов. В такой ситуации возникает реальная проблема конфликта имен. С одной стороны, желательно использовать наглядные имена, которые облегчают понимание, отладку и документирование кода. С другой стороны, если классов сотни и тысячи, то неизбежны совпадения имен классов.

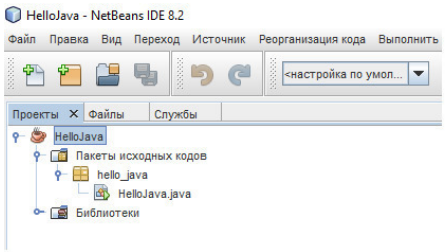

Рис. 2.6 Структура программы на языке Java

Для устранения возможных конфликтов имен классов и четкого структурирования проекта применяется разбиение на пакеты.

С физической точки зрения пакет Java – это отдельный каталог (папка) на диске компьютера. Имя каталога совпадает с именем пакета. Например, если вы установили NetBeans IDE на компьютер с ОС Windows с настройками по умолчанию, то в папке **Documents** будет создана папка **NetBeansProjects**. В ней расположены папки проектов. Сейчас там появилась папка **HelloJava**, внутри нее находится папка **src**. Она соответствует папке «Пакеты исходных кодов» на рис. 2.6. Внутри нее находится собственно папка пакета hellojava, которая содержит файл HelloJava. java. Как видите, физическая структура каталогов повторяет структуру проекта, отображаемую в окне NetBeans IDE.

При разработке простого приложения имя пакета можно не указывать, и среда NetBeans автоматически создаст безымянный «пакет по умолчанию». Но лучше сразу привыкать к использованию именованных пакетов.

Каждый пакет формирует отдельное *пространство имен*. Это важно для крупных профессиональных разработок, когда один и тот же пакет может быть включен в состав различных независимых проектов. Благодаря разделению классов по пакетам, разработчики застрахованы от случайных конфликтов имен.

В первой строке листинга 2.1 мы указали, что создаем пакет hello_java и работаем в его пространстве имен. Допускается создание подпакетов (вложенных пакетов). В таком случае имя пакета и подпакета разделяется точкой:

```
package main_pack.sub_pack;
```

Глубина вложенности пакетов формально не ограничена. Физическая структура файлов и папок на диске компьютера должна соответствовать структуре вложенных пакетов проекта.

Чтобы воспользоваться в программе классами из стороннего пакета, его нужно *импортировать* при помощи инструкции import. После нее указывают имя пакета и, через точку, имя импортируемого класса или звездочку *, если импортируются все публичные классы пакета:

```
import mypack.MyClass;
import nextpack.*;
```

Использование звездочки не увеличит размер приложения, потому что компилятор все равно включит в него только нужные классы. Но если пакет содержит несколько сотен или тысяч классов, то время компиляции может заметно возрасти.

Теперь разверните в окне просмотра проекта (рис. 2.6) папку «Библиотеки». По умолчанию там находится главный системный пакет JDK, который содержит предоставленные разработчиком классы для работы с системой. Этот пакет всегда подключен на уровне среды разработки, поэтому в явном импорте классов SDK нет нужды.

Теперь мы можем сказать, что означает строка из листинга 2.1:

```
System.out.println («Hello Java»);
```

В этой строке мы последовательно обращаемся к встроенному классу System, его полю out и методу println (String). Компилятор преобразует эту строку в байт—код, который заставит виртуальную Java—машину вывести в окно терминала строку текста.

При разработке собственных приложений вы можете подключать к проекту библиотеки сторонних разработчиков. Например, чтобы работать с последовательными портами компьютера, можно воспользоваться библиотекой JSSC, а для работы с базами данных MS Access пригодится библиотека UCanAccess.

Глава 3. Переменные и операторы

Вы получили общее представление о языке Java. Теперь настало время перейти к более конкретным понятиям. В этой главе будет рассказано о переменных, типах данных и операторах.

3.1 Переменные и типы данных

Может показаться, что в программировании нет ничего проще, чем переменная. Какие могут быть сложности? Тем не менее, для начинающих программистов сложности есть. Неправильное понимание того, как устроен мир переменных и данных, может привести к появлению трудно локализуемых ошибок.

Переменная представляет собой *указатель* на физическую область памяти, в которой хранятся данные. При помощи указателя мы можем записывать значения в память и считывать их оттуда. Иными словами, переменная – это имя фрагмента памяти компьютера. Размер этого фрагмента зависит от того, какие данные мы собираемся хранить.

Разрабатывая или запуская программу, мы не знаем заранее, по каким физическим адресам будут находиться данные в конкретном компьютере. Более того, в компиляторах современных языков принимаются специальные меры для дополнительного сокрытия информации о физическом размещении данных. Это делается для того, чтобы злоумышленнику было труднее получить доступ к критически важным данным, анализируя содержимое оперативной памяти компьютера.

До первого обращения к переменной ее надо объявить. При объявлении переменной указывают ее тип и имя. Это важно, потому что компилятор должен заранее знать, какой объем памяти выделить для хранения переменной, и как истолковывать данные, прочитанные из памяти.

Типы данных в языке Java можно разделить на две основные категории: *примитивные* (простые) и *ссылочные*. Они различаются по способу размещения данных в памяти.

Данные примитивного типа хранятся непосредственно в той ячейке памяти, которая ассоциирована с именем переменной. Обращаясь к переменной по имени, мы тем самым, обращаемся к данным в памяти. Если вы сравниваете две переменных, то сравниваются *данные*, которые с ними связаны. Если вы присваиваете одной переменной примитивного типа значение другой переменной примитивного типа, то происходит копирование *данных*.

В случае использования ссылочного типа в ячейке памяти, которая ассоциирована с именем переменной, хранится адрес данных, т.е. ссылка на данные, а не сами данные.

Необходимость в ссылочном типе данных можно продемонстрировать с помощью простого примера. Допустим, вы объявили *строковую переменную* с начальным значением «Java». Под это значение выделяется место в памяти. В процессе работы программы этой переменной присваивается новое значение «Hello, World!». Очевидно, что это совершенно другой объем данных, который не поместится в ранее отведенном фрагменте памяти.

Иными словами, ссылочные типы предназначены для работы с динамически создаваемыми и уничтожаемыми сущностями, объем которых невозможно предсказать заранее.

В таком случае программа размещает новые данные в другом фрагменте памяти. Новый адрес этих данных записывается в ячейку, которая ассоциирована с переменной ссылочного типа. Если на старые данные больше не ссылается ни одна переменная, то они превращаются в *мусор* (garbage) и удаляются из памяти при помощи специального *сборщика мусора* (garbage collector). В языке Java сборка мусора выполняется автоматически.

При проверке ссылочных переменных на равенство сравниваются не сами данные, а их *адреса*, хранящиеся в ссылочных переменных. Если вы присваиваете одной ссылочной переменной значение другой ссылочной переменной, при этом копируется адрес данных, а не сами данные.

3.1.1 Примитивные типы данных

В языке Java заявлено восемь примитивных типов данных. Первые четыре используются для хранения целых чисел.

byte – однобайтное целое – предназначен для хранения целых чисел в диапазоне от -128 до 127 и занимает один байт в памяти.

short – короткое целое – занимает два байта в памяти и применяется для хранения чисел в диапазоне от -32768 до32767.

int – целое – занимает 4 байта в памяти и применяется для хранения чисел в диапазоне от -2^{31} (-2147483648) до 2^{31}—1 (2147483647). Это стандартный тип данных для работы с целыми числами.

При работе с числовыми данными старайтесь использовать тип int. Это связано с особенностями автоматического приведения типов, а также с тем, что целочисленные литералы (например, 10 или 123) в коде программы обрабатываются компилятором, как тип int. Приведение типов мы обсудим далее в этой главе.

long – длинное целое – занимает 8 байтов в памяти и хранит числа в диапазоне от -2^{63} до 2^{63}—1. На практике настолько большие числа встречаются редко. Чтобы определить длинное целое число, следует добавить суффикс «L» в конце, например 5201225834L.

В дополнение к целочисленным типам, имеется два типа данных для хранения чисел с плавающей точкой.

float – с плавающей точкой – занимает 4 байта в памяти и может хранить числа в диапазоне от $-3{,}4\times10^{38}$ до $3{,}4\times10^{38}$ с дискретностью $3{,}4\times10^{-38}$. Такая точность представления соответствует 7 знакам после запятой. Если вы попытаетесь сохранить в типе float число 1,234567891 (10 знаков), оно будет округлено до 1,234568 (7 знаков).

Что такое дискретность? Вы не можете задать значение типа float с произвольной точностью. Ведь количество байт памяти для хранения этого числа ограничено. Если мы начнем перечислять подряд, начиная с ноля, числа с плавающей точкой, то они будут следовать с некоторым шагом (дискретностью) в младших разрядах: 0; $3{,}4\times10^{-38}$; $6{,}8\times10^{-38}$ и т. д. Величину дискретности можно условно назвать погрешностью представления числа. Для достижения более высокой точности применяется тип double.

double – с плавающей точкой, двойной точности – занимает 8 байтов в памяти и может хранить числа в диапазоне от $-1{,}7\times10^{308}$ до $1{,}7\times10^{308}$ с дискретностью $1{,}7\times10^{-308}$. Если вы не скованы ограничениями объема памяти, используйте тип double вместо float, как более точный.

По умолчанию, как только вы использовали десятичную точку в программе на языке Java, этому значению присваивается тип double. Если вы хотите, чтобы это число было истолковано именно как float, добавьте суффикс «F» в конце числа.

Кроме шести перечисленных типов, Java располагает двумя специфическими типами данных.

char – символ – занимает 2 байта и применяется для хранения одиночного символа Unicode, например «A», "@», «$» и т. д.

boolean – логический – это особый тип данных, который может хранить только два фиксированных значения: true (истина) и false (ложь). Размер занятой памяти зависит от реализа-

ции Java—машины. Этот тип данных широко используется в условных операторах и операторах цикла, которые мы рассмотрим позже.

Все остальные типы данных, включая пользовательские типы, являются ссылочными.

3.1.2 Объявление и инициализация переменных

При объявлении переменной указывается тип переменной и ее имя. Переменная может быть объявлена в любом месте программы, главное – до первого использования.

```
boolean fileSaved;
```

Если объявляется несколько переменных одного типа, то их можно перечислить через запятую.

```
int userNum, userAge, userWeight;
```

Одновременно с объявлением переменной ей можно присвоить значение. Эта процедура называется инициализацией.

```
int start=10, end=100;
```

Допускается динамическая инициализация переменной, когда ей присваивается значение, полученное вычислением из значений других переменных. Исходные переменные должны быть объявлены и инициализированы ранее.

```
int start=5, end=10;
int sum=a+b;
```

В этом примере переменная sum инициализирована значением 15.

Обратите внимание, что в момент динамической инициализации не возникает связь между переменными. Например, если после инициализации изменится значение переменных start и end, это никак не повлияет на значение sum.

3.1.3 Доступность переменных

Доступность, или область видимости переменных – это важный аспект программирования. Если кратко, переменная доступна внутри блока, определенного парой фигурных скобок, внутри которого она объявлена. Например, если переменная объявлена внутри цикла, то она будет доступна только внутри этого цикла. Снаружи цикла может быть объявлена переменная с таким же именем, но фактически это будет совершенно другая переменная.

Допустим, в некой фирме работает Иванов, он выполняет свои задачи в пределах штата фирмы. В соседнем офисе тоже работает Иванов, но это другой человек, который делает другую работу. Директор первой фирмы не может отдавать распоряжения Иванову из второй фирмы. Для него второй Иванов недоступен.

Если переменная доступна только внутри некого метода (функции), то она называется *локальной*. Если переменная задана на уровне класса, она называется *глобальной*. Глобальные переменные обычно доступны любому из методов, входящих в класс. При использовании глобальных переменных необходимо соблюдать осторожность. Если внутри одного из методов случайно изменить значение глобальной переменной, другие методы будут получать неправильное значение. Это приведет к появлению трудно локализуемой *логической ошибки*, на которую не реагирует компилятор.

3.1.4 Ввод и считывание данных

Переменным можно присваивать значения, введенные извне. Давайте немного отвлечемся от абстрактных рассуждений и запустим две простых программы, которые запрашивают

данные у пользователя и обрабатывают их. К этому моменту вы должны уметь создавать проекты в среде NetBeans IDE, поэтому я привожу только исходный код примеров.

Программа из листинга 3.1 поддерживает консольный ввод – пользователь читает запросы программы и вводит данные в окне системного монитора среды NetBeans. В программе из листинга 3.2 задействованы модальные окна с привычным графическим оформлением. Вы увидите, насколько просты эти программы. Не волнуйтесь, если что-то непонятно. Пока просто привыкайте к новым терминам. По мере чтения этой книги придет полное понимание.

Листинг 3.1 Чтение консольного ввода, вывод в консоль

```
import java.util.Scanner;

public class Listing3_1 {

public static void main (String [] args) {
// Создаем объект input класса Scanner
Scanner input = new Scanner(System.in);
// Переменная для хранения имени пользователя
String name;
// Переменная для хранения отчества пользователя
String surName;
// Переменная для хранения даты рождения пользователя
int yearBorn;
// Переменная для хранения текущего года
int yearNow;
// Выводим запрос данных
System.out.print («Ваше имя:»);
// Считываем имя (строка)
name = input.nextLine ();
System.out.print («Ваше отчество:»);
// Считываем отчество (строка)
surName = input.nextLine ();
System.out.print («Какой сейчас год?»);
// Считываем текущий год (целое число)
yearNow = input.nextInt ();
System.out.print («В каком году вы родились?»);
// Считываем год рождения (целое число)
yearBorn = input.nextInt ();
System.out.println («Здравствуйте, "+name+" "+surName+»!»);
System.out.println («Ваш возраст: "+ (yearNow-yearBorn) +».»);

}
}
```

В первой строке этой программы мы импортируем класс Scanner, который входит в состав системного пакета java. util. Затем мы создаем новый объект класса Scanner и назначаем ему идентификатор (имя) input. После этого приступаем к получению данных от пользователя. Выводим в консоль текстовый запрос и считываем ответ. Обратите внимание, что текстовые ответы мы считываем при помощи метода nextLine (), а целочисленные при помощи

метода nextInt (). В противном случае возникнет ошибка несоответствия типа данных. Ведь мы объявили переменные yearNow и yearBorn как целые числа.

Отдельно разберем строку

System.out.println («Ваш возраст: "+ (yearNow-yearBorn) +».»);

В этой строке происходит арифметическое вычисление возраста пользователя, формирование строки вывода и вывод в консоль. Выражение (yearNow-yearBorn) обязательно должно быть в круглых скобках, потому что сначала должно быть вычислено его значение, а затем результат вычисления будет преобразован из числа в строку (автоматическое приведение типов).

Наберите или скачайте исходный код программы и запустите проект на выполнение. Введите ответы на вопросы. В окно терминала должно быть выведено что-то наподобие этого:

```
run:
Ваше имя: Иван
Ваше отчество: Петрович
Какой сейчас год? 2018
В каком году вы родились? 1988
Здравствуйте, Иван Петрович!
Ваш возраст: 30.
СБОРКА УСПЕШНО ЗАВЕРШЕНА (общее время: 22 секунды)
```

На компьютере с ОС Windows вместо символов кириллицы вы можете увидеть квадратики. В этом случае необходимо настроить кодировку проекта. В окне просмотра содержимого проекта щелкните правой кнопкой мыши на названии проекта и выберите пункт **Свойства** контекстного меню. В открывшемся окне найдите поле **«Кодировка»** и выберите в списке кодировку **windows—1251**. Нажмите **OK**.

Вторая программа имеет графический интерфейс, основанный на *модальных окнах*. Это специальные окна, которые содержат сообщение или поле ввода. Чтобы программа продолжила выполнение, пользователь обязательно должен отреагировать на появление окна – ввести данные или прочитать сообщение и закрыть.

Листинг 3.2 Ввод и вывод данных в модальных окнах

```
// импортируем класс JOptionPane из библиотеки Swing
import javax.swing.JOptionPane;
public class Listing3_2 {
public static void main (String [] args) {
// Объявление числовых переменных
int yearNow, yearBorn, userAge;
// Объявление строковой переменной
String userData;
// Выводим окно запроса текущей даты
userData = JOptionPane.showInputDialog («Какой сейчас год?»);
// Преобразуем строку в число в явном виде
yearNow = Integer.parseInt (userData);
// Выводим окно запроса года рождения
userData = JOptionPane.showInputDialog («В каком году вы родились?»);
// Преобразуем строку в число в явном виде
yearBorn = Integer.parseInt (userData);
// Вычисляем возраст
```

```
userAge = yearNow – yearBorn;
// Выводим окно сообщения с результатом
JOptionPane.showMessageDialog (null, «Ваш возраст: " + userAge);
}
}
```

В первой строке программы мы импортируем класс JOptionPane из библиотеки Swing. Библиотека Swing содержит набор классов для разработки приложений с графическим интерфейсом. Это очень емкая и мощная библиотека, входящая в пакет поставки SDK. Вы будете постоянно использовать ее при разработке приложений с графическим интерфейсом. Класс JOptionPane предназначен для создания стандартных модальных (диалоговых) окон. Для вывода окна с запросом данных применяется метод showInputDialog (), а для вывода сообщения – метод showMessageDialog ().

Любые значения, возвращаемые методом showInputDialog () являются строковыми данными. Чтобы выполнить над ними арифметические действия, необходимо в явном виде преобразовать строки в числа. Мы делаем это при помощи метода parseInt () системного класса Integer:

```
yearNow = Integer.parseInt (userData);
```

Программа завершается вычислением возраста пользователя и выводом результата.

Запустите проект на выполнение. Вы должны поочередно увидеть три диалоговых окна (рис. 3.1).

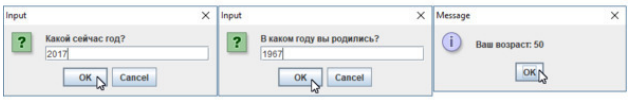

Рис.3.1 Диалоговые окна запроса и вывода данных

Если все работает правильно, нажмите клавишу **F11** или выберите пункт меню **Выполнить | Собрать проект**. Будет создан исполняемый файл приложения. Его можно запустить на любом компьютере, где установлена Java-машина. Оформление окон приложения – цветовая схема, форма кнопок – может различаться в зависимости от операционной системы и реализации Java-машины.

По умолчанию файл проекта находится в папке Документы | NetBeansProjects. Внутри папки с именем проекта найдите папку dist. В этой папке находится готовый распространяемый файл приложения с расширением jar.

3.2 Приведение типов

Иногда возникает ситуация, когда в одном выражении присутствуют разные типы данных. Будем считать, что это осознанное действие, а не ошибка программиста, ибо такие ошибки чрезвычайно коварны. Несоответствие типов в выражении не всегда влечет за собой ошибку компиляции, но программа может вести себя не так, как ожидалось. Изменение типа данных в процессе выполнения программы называется *приведением типа*. Реализация приведения типов зависит от конкретного языка. Некоторые языки допускают большие вольности в приведении типов. За это их резко критикуют профессиональные программисты.

Про особенности приведения типов в разных языках программирования можно написать отдельную брошюру. Но в нашем вводном курсе мы ограничимся изложением основных принципов.

Приведение типов разделяется на *явное* (указанное программистом в коде) и *неявное* (*автоматическое*).

При явном приведении типов перед значением или выражением в скобках указывается новый тип, например:

```
double x = 15.7;
y = (int) 15.7;
```

В этом примере число с плавающей точкой приводится к типу «целое», при этом просто отбрасывается дробная часть. Результатом приведения будет усеченное значение 15, а не округленное 16.

Обратное преобразование из целого в число с плавающей точкой тоже выполняется, но это происходит автоматически. Запомните простое правило: *если в выражении участвуют операнды разных типов, то результат приводится к тому типу, который занимает больше места в памяти*. Поэтому важно, чтобы тип переменной, которой вы хотите присвоить результат вычислений, совпадал с типом результата. Вот простой пример приведения типов:

```
byte a = 2;
a = (byte) (a*5);
```

В этом примере целочисленный литерал 5 трактуется, как значение типа int, поэтому результат умножения будет расширен до типа int. Но переменная объявлена, как byte, поэтому возникнет конфликт выделения памяти и ошибка компиляции.

Чтобы избежать ошибки, мы в явном виде приводим результат умножения к типу byte. При этом из 32 байт остаются только младшие 8, а остальные отбрасываются. Это опасная потеря информации. Может получиться так, что при маленьких исходных значениях результат будет верным. Но стоит разрядности результата умножения превысить 8 битов, и после приведения типов вы получите неправильный результат вычислений. Такая блуждающая ошибка зависит от сочетания факторов и трудно поддается локализации в коде.

Автоматическое приведение типов часто применяется при суммировании строки и числа. В этом случае число автоматически преобразуется в строку и выполняется обычная конкатенация (слияние) строк. Например:

```
int yearNow = 2018;
System.out.println («Текущий год " + yearNow);
```

В окно терминала будет выведена строка «Текущий год: 2018».

Обратное преобразование из строки в число автоматически не выполняется. Необходимо воспользоваться специальными методами, такими как Integer.parseInt (), Double.parseInt () и т. п. в зависимости от нужного типа. В листинге 3.2 вы уже встречали преобразование из строки в число.

3.3 Основные операторы

Основные операторы языка Java можно разделить на четыре группы: арифметические, логические, битовые и операторы сравнения.

По количеству обязательных операндов в выражении операторы разделяются на унарные (один операнд), бинарные (два операнда) и тернарные (три операнда).

3.3.1 Арифметические операторы

К арифметическим операторам относятся сложение (+), вычитание (-), умножение (*), деление (/), вычисление остатка (%), инкремент (++) и декремент (– -).

Допустим, мы задали значения x=18 и y=4. Тогда результаты использования операторов будут выглядеть так:

Сложение: x + y = 22

Вычитание: x – y = 14

Умножение: x*y = 72

Пока ничего необычного, но дальше будет немного сложнее.

Деление: 18 / 4 = 4

Неожиданно, не так ли? В языке Java результат деления одного целого числа на другое целое число будет целочисленным, остаток отбрасывается без округления. Получить результат деления с дробной частью можно двумя способами: объявить один или оба операнда как число с плавающей точкой или использовать явное приведение.

```
18 / 4.0 = 4.50
(double) 18/4 = 4.50
```

Вычисление остатка: 18%4 = 2. При делении 18/4 нацело мы получаем частное 4 (4*4=16) и остаток 2 (18—16=2). Иными словами, остаток – это побочный продукт целочисленного деления.

Инкремент: оператор постфиксного инкремента x++ сперва возвращает исходное значение переменной, затем увеличивает его на единицу. Оператор префиксного инкремента ++x сперва увеличивает значение переменной на 1, затем возвращает новое значение.

Строка с постфиксным инкрементом

```
System.out.print (x++);
```

равнозначна последовательности команд

```
System.out.print (x);
x = x +1;
```

Строка с префиксным инкрементом

```
System.out.print (++x);
```

равнозначна последовательности команд

```
x = x +1;
System.out.print (x);
```

Декремент: оператор постфиксного декремента сперва возвращает исходное значение переменной, затем уменьшает его на единицу. Оператор префиксного декремента сперва уменьшает значение переменной на 1, затем возвращает новое значение.

Строка с постфиксным декрементом

```
System.out.print (x – -);
```

равнозначна последовательности команд

```
System.out.print (x);
x = x – 1;
```

Строка с префиксным декрементом

```
System.out.print (– – x);
```

равнозначна последовательности команд

```
x = x – 1;
```

```
System.out.print (x);
```

3.3.2 Логические операторы

Логические операторы предназначены для использования с логическими операндами и создания условий для логических операторов.

Логическое И (&) – результатом выражения A&B является true, если оба операнда имеют значение true. Если хотя бы один из операндов имеет значение false, то результатом является false.

Укороченное логическое И (&&) – выражение A&&B вычисляется точно так же, как A&B, но если при проверке операнда A оказывается, что оно равно false, то значение B уже не проверяется, а сразу возвращается значение false.

Логическое ИЛИ (|) – результатом выражения A|B является true, если значение хотя бы одного из операндов является true. В ином случае возвращается значение false.

Укороченное логическое ИЛИ (||) – результат выражения A||B совпадает с результатом A|B, но если при проверке операнда A оказывается, что он имеет значение true, то второй операнд не проверяется, и сразу возвращается значение true.

Логическое исключающее ИЛИ (^) – результатом выражения A^B является true, если один операнд имеет значение true, а другой имеет значение false. Если оба операнда одновременно имеют значение true, или оба операнда одновременно имеют значение false, то возвращается значение false.

Унарное логическое отрицание (!) – результатом выражения! A является false, если операнд имеет значение true, и наоборот.

При помощи логических операторов можно формировать сложные выражения с участием нескольких операндов, например:

A&B&C – это выражение возвращает значение true, только если все три операнда одновременно имеют значение true.

A|B|C – это выражение возвращает true, если хотя бы один из операндов имеет значение true.

A&B|C – это выражение возвращает true, если A и B одновременно имеют значение true, или C имеет значение true. Оператор & имеет более высокий приоритет, поэтому сначала вычисляется значение выражения A&B, и результат вступает в логическую операцию ИЛИ с операндом C.

3.3.3 Битовые операторы

Битовые (или побитовые) операторы предназначены для операций с целыми числами на уровне их побитового представления.

Битовое И (&) – выражение A&B выполняется побитово, т.е. отдельно для каждого разряда. Если оба бита единичные, то в соответствующем разряде результата будет единица. Если хотя бы один из битов нулевой, в разряд результата записывается ноль.

Пример: 1101 & 0110 = 0100

Битовое ИЛИ (|) – выражение A|B выполняется побитово. Если хотя бы один из битов единичный, то в соответствующий разряд результата будет записана единица. Если оба бита нулевые, то в разряд результата будет записан ноль.

Пример: 1101 | 0110 = 1111

Битовое исключающее ИЛИ (^) – выражение A^B выполняется побитово. Если один из сравниваемых битов нулевой, а другой единичный, то в разряд результата записывается единица. Если оба бита нулевые, или оба бита единичные, то в разряд результата записывается ноль.

Пример: 1101 ^ 0110 = 1011

Битовый сдвиг вправо (>>) – результатом выполнения оператора A>> n является число, которое получилось сдвигом двоичного числа A вправо на n позиций. При сдвиге сохраняется знак числа, то есть младшие разряды теряются, а старшие заполняются содержимым знакового бита (0 для положительных чисел и 1 для отрицательных).

Примеры: (11010010)>> 2=11110100, (01010010)>> 2=00010100

Беззнаковый битовый сдвиг вправо (>>>) – результатом выполнения оператора A>>> n является число, которое получилось сдвигом двоичного числа A вправо на n позиций. При сдвиге НЕ сохраняется знак числа, то есть младшие разряды теряются, а старшие заполняются нулями.

Битовый сдвиг влево (<<) – результатом выполнения оператора A <<n является число, которое получилось сдвигом двоичного числа A влево на n позиций. При этом старшие разряды теряются, а младшие дополняются нулями.

3.3.4 Операторы сравнения

Если условие, заданное оператором сравнения, выполняется, то выражение возвращает значение true. В противном случае возвращается значение false. Все операторы сравнения бинарные – содержат только два операнда.

Равно (==) – выражение A==B возвращает true, если значение операнда A равно значению операнда B. Обратите внимание, оператор сравнения состоит из двух знаков равенства. Если вы используете одиночный знак равенства, то получится не сравнение, а присвоение значения. Среда NetBeans предупредит вас о возможной ошибке, хотя с формальной точки зрения это логическая, а не синтаксическая ошибка.

Не равно (!=) – выражение A!=B возвращает true, если значение операнда A отлично от значения операнда B.

Конец ознакомительного фрагмента.