

Программирование

на

JAVA и C#

О. В. Герман
Ю. О. Герман

ДЛЯ
СТУДЕНТА



+ CD



УДК 681.3.06
ББК 32.973.26-018.1
Г38

Герман О. В., Герман Ю. О.

Г38 Программирование на Java и C# для студента. — СПб.: БХВ-Петербург, 2005. — 512 с.: ил.

ISBN 5-94157-710-9

Рассмотрены основные вопросы программирования на языках JAVA и C#, включая их сравнительное описание как двух важнейших и весьма сходных прикладных платформ для создания современных сетевых приложений. Книга содержит теоретическую часть, объясняющую основные моменты программирования, и практическую, включающую задания, контрольные вопросы и много законченных примеров с подробными объяснениями и комментариями, которые позволяют эффективно перейти к самостоятельному написанию программ на языках JAVA и C#. На компакт-диске размещены листинги примеров, рассмотренных в книге.

Для студентов, преподавателей и программистов

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Дарья Масленникова</i>
Компьютерная верстка	<i>Татьяны Олоновой</i>
Корректор	<i>Наталья Першакова</i>
Дизайн обложки	<i>Игоря Цырульников</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 28.09.05.

Формат 60×90^{1/16}. Печать офсетная. Усл. печ. л. 32.

Тираж 3000 экз. Заказ №

"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953 Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-710-9

© Герман О. В., Герман Ю. О., 2005
© Оформление, издательство "БХВ-Петербург", 2005

Оглавление

Введение.....	9
Сравнительный анализ Java и C#: общность платформ и отличительные особенности.....	11
Зачем нужно знать и Java и C#?	15
 ЧАСТЬ I. JAVA	17
Глава 1. Основы программирования на языке Java	19
Программирование "без классов"	23
Классы	40
Объявление переменных и методов, использование их в программе.....	52
Создание визуального интерфейса	63
Программирование обработки событий от элементов, мышь и клавиатуры	67
Программирование ввода-вывода с использованием файлов	78
Работа со строками	89
Использование массивов.....	91
Апплеты.....	93
Обработка исключительных ситуаций.....	96
Работа с графикой.....	98
 Глава 2. Практические занятия по Java	109
Основы HTML.....	109
Цель занятия	109
Краткие теоретические сведения.....	109

Задание.....	119
Контрольные вопросы	119
Использование скриптов JavaScript в документах HTML.....	120
Цель занятия	120
Краткие теоретические сведения.....	120
Задание.....	132
Контрольные вопросы	133
Введение в Java.....	133
Цель занятия	133
Краткие теоретические сведения.....	134
Задание.....	144
Контрольные вопросы	145
Реализация взаимодействия между апплетами.....	145
Цель занятия	145
Краткие теоретические сведения.....	146
Задание.....	155
Контрольные вопросы	155
Внутренняя база данных апплета.....	156
Цель занятия	156
Краткие теоретические сведения.....	156
Задание.....	163
Контрольные вопросы	164
Работа с формами и меню	164
Цель занятия	164
Краткие теоретические сведения.....	164
Задание.....	174
Контрольные вопросы	174
Java и базы данных	175
Цель занятия	175
Краткие теоретические сведения.....	175
Задание.....	183
Контрольные вопросы	184
Основы XML. Преобразование XML-HTML.	
Использование JavaScript	184
Цель занятия	184
Краткие теоретические сведения.....	184
Задание.....	192
Контрольные вопросы	192
Взаимодействие XML-Java-JavaScript	193
Цель занятия	193
Краткие теоретические сведения.....	193

Задание.....	203
Контрольные вопросы	203
Чтение XML-файла с использованием файлового диалога	203
Цель занятия	203
Краткие теоретические сведения.....	203
Задание.....	212
Контрольные вопросы	213
Потоки в Java.....	213
Цель занятия	213
Краткие теоретические сведения.....	213
Задание.....	221
Контрольные вопросы	222
Создание приложений "клиент-сервер"	222
Цель занятия	222
Краткие теоретические сведения.....	223
Задание.....	235
Дополнительные сведения.....	236
Контрольные вопросы	246
Доступ к серверной базе данных из клиента.....	247
Цель занятия	247
Краткие теоретические сведения.....	247
Задание.....	275
Контрольные вопросы	276
Использование Java Beans в других средах	277
Цель занятия	277
Краткие теоретические сведения.....	277
Задание.....	287
Контрольные вопросы	288
Изучение механизма сериализации	288
Цель занятия	288
Краткие теоретические сведения.....	288
Задание.....	300
Контрольные вопросы	300
Создание сервлетов.....	300
Цель занятия	300
Краткие теоретические сведения.....	301
Задание.....	307
Контрольные вопросы	308

Создание почтовой службы в стандартном Java.....	308
Цель занятия	308
Краткие теоретические сведения.....	309
Задание.....	312
Контрольные вопросы	312
Создание JSP-страниц.....	313
Цель занятия	313
Краткие теоретические сведения.....	313
Задание.....	316
Контрольные вопросы	316
Создание простого браузера	317
Цель занятия	317
Краткие теоретические сведения.....	317
Задание.....	323
Контрольные вопросы	323
Сводка основных использованных команд Java.....	324
 ЧАСТЬ II. C#	331
 Глава 3. Основы программирования на языке C#	333
Введение в язык C#	333
Платформа C# для Java-программистов.....	342
Программирование "без классов".....	345
Использование классов.....	350
Использование подпрограмм.....	362
Объявление массивов	367
Работа с файлами	371
Сериализация объектов.....	376
Создание приложений на основе формы.....	379
Работа со строками	386
Создание сборок.....	389
Создание Web-приложений	392
Реализация API-вызовов	398
 Глава 4. Практические занятия по C#	403
Файловый ввод-вывод в C#	403
Цель занятия	403
Краткие теоретические сведения.....	403

Задание.....	417
Контрольные вопросы	419
Работа с базами данных в С#	419
Цель занятия	419
Краткие теоретические сведения.....	419
Задание.....	426
Контрольные вопросы	426
Простейшее рисование в С#	426
Цель занятия	426
Краткие теоретические сведения.....	427
Задание.....	433
Контрольные вопросы	433
Изучение механизма потоков для смены графических изображений	433
Цель занятия	433
Краткие теоретические сведения.....	433
Задание.....	443
Контрольные вопросы	443
Создание собственных компонентов.....	444
Цель занятия	444
Краткие теоретические сведения.....	444
Задание.....	452
Контрольные вопросы	453
Клиент-серверное взаимодействие на основе протоколов TCP и HTTP	453
Цель занятия	453
Краткие теоретические сведения.....	453
Задание.....	461
Контрольные вопросы	461
Работа с классом таймера	461
Цель занятия	461
Краткие теоретические сведения.....	462
Задание.....	468
Контрольные вопросы	469
Обработка польской записи.....	469
Цель занятия	469
Краткие теоретические сведения.....	469
Задание.....	480
Контрольные вопросы	481

Работа с коллекциями	481
Цель занятия	481
Краткие теоретические сведения.....	481
Задание.....	491
Контрольные вопросы	491
Сводка основных использованных команд C#	491
Приложение. Описание компакт-диска	499
Содержимое компакт-диска.....	499
Инструкции по работе с листингами программ на Java и C#	499
Запуск классов Java.....	500
Запуск апплетов Java.....	500
Запуск приложений C#	502
Установка Tomcat и Java	503
Список литературы	505
Предметный указатель	507



Глава 1

Основы программирования на языке Java

Язык Java — это объектно-ориентированный язык, прототипом которого можно считать C++. Java в значительной степени упростил написание программ и позволил выполнять их на разных платформах (под управлением различных операционных систем). Простота этого языка, его хорошая защищенность и возможности, предоставляемые для программирования в Интернете, сделали его весьма популярным.

Java является языком сетевого программирования, в первую очередь ориентированным на среду Интернет. Концепция программирования в Интернете базируется на рассмотрении двух сторон: стороны *клиента* и стороны *сервера*. К серверу обращаются многие клиенты. Обычно обращение происходит за информацией из базы данных либо для выполнения некоторой полезной программы. Клиентом является подключенный к Интернету конечный пользователь. Такой пользователь, работая на своем индивидуальном компьютере, должен знать интернет-адрес сервера и имя той программы, с которой он хочет связаться. Для доступа к серверу клиент должен создать свой сайт на языке HTML или построить *апплет* на языке Java и ввести в него функциональную часть, выполняющую обращение к серверу. Посредником между клиентской и серверной сторонами является специальная программа — *браузер*. Содержимое сайта браузер передает на сервер. На сервере выполняется программа *Web-сервер*, которая взаимодействует с

браузером через определенный порт и активизирует обработчик скрипта или сервлет (если приложение сервера, обрабатывающее сайт, написано на языке Java). Обработчик скрипта выполняет конкретную прикладную задачу, используя принятые данные сайта. Например, обработчик скрипта может обратиться к базе данных, передав в свою очередь запрос серверу баз данных, размещенному, как правило, на той же машине, что и Web-сервер. Результаты своей работы (как правило, так же в форме сайта) обработчик скрипта или сервлет возвращает клиенту.

Внешне после запуска сайт может иметь, например, такой вид (рис. 1.1).

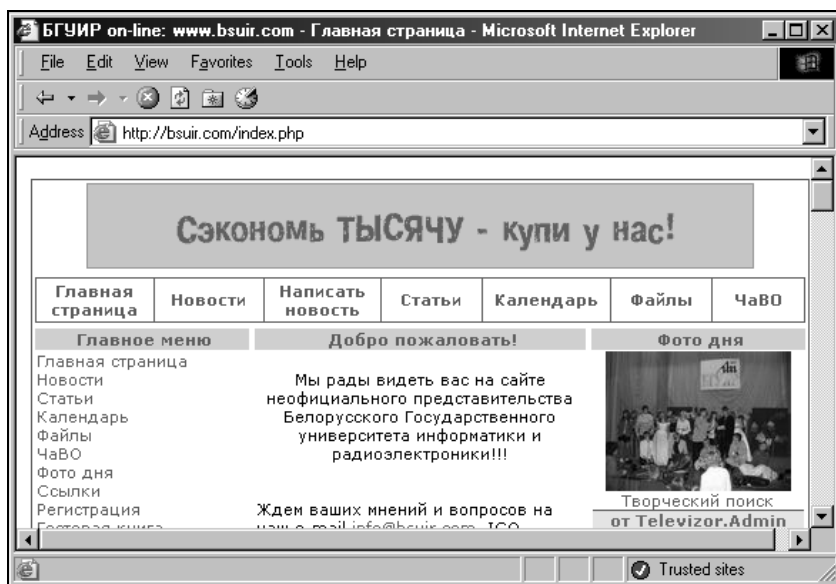


Рис. 1.1. Неофициальный сайт Белорусского государственного университета информатики и радиоэлектроники (Минск)

Поскольку язык HTML используется для программирования сайтов очень широко, ему посвящено наше первое практическое занятие. На сайте можно разместить много самой разнообразной информации. Эту информацию можно передать серверному приложению средствами специального посредника (браузера) —

например, Internet Explorer. Браузер связывается с Web-сервером по каналу связи через определенный порт (как правило, с номером 8080) и передает ему информацию в определенном формате (протоколе). Обычно таким протоколом является HTTP (Hypertext Transfer Protocol, протокол передачи гипертекстовых файлов). В сайты, написанные на HTML, можно вставлять *апплеты (applets)* — динамические окна, запрограммированные средствами языка Java. Сервер, приняв документ от браузера, передает управление серверному приложению — обработчику скрипта или сервлету, который может получить значения элементов формы — текстовых полей, выделенных элементов списков, переключателей. Типичная задача серверного приложения — зайти в базу данных и найти информацию, затребованную клиентом. Эта задача возникает не только в Интернете, но и при работе в локальных сетях, например, в условиях фирмы, учреждения или производства. Таким образом, создание распределенных приложений на Java является одним из основных назначений данного языка.

Перед тем как перейти к описаниям практических занятий, мы предположим им общие теоретические сведения по языку Java.

Основными *предварительными* моментами в освоении этого языка являются следующие:

- ☐ использование классов и классовых переменных как основных структурных единиц программ;
- ☐ объявление переменных и методов, их использование в программе;
- ☐ создание визуального интерфейса (форм, кнопок, списков, меню и пр.);
- ☐ программирование обработки событий от элементов, мыши и клавиатуры;
- ☐ программирование ввода/вывода через файлы и т. д.

Последующее изложение языка включает более сложные вопросы, например, использование потоков, клиент-серверных технологий, работу с XML (eXtended Markup Language, расширенный язык разметки) и пр. Вам потребуется установить у себя Java SDK 1.x (например, $x = 4$). Эту систему можно бесплатно скачать через

Интернет с сервера фирмы Sun, расположенного по адресу <http://Java.sun.com/j2se/1.4/download-windows.html>, либо последнюю версию J2SE SDK (Java 2 Standard Edition Software Development Kit), которую можно скачать с сайта фирмы Sun по адресу <http://Java.sun.com/j2se/>. Основными программами, которые задействует этот практикум, являются `java.exe` и `javac.exe`. Последняя из этих программ осуществляет создание классов путем компиляции исходных Java-файлов. Программа `java.exe` выполняет созданные классы. Java-классы выполняются на различных платформах: Windows, UNIX, Linux, Solaris и др. при условии, что на компьютере установлена виртуальная машина Java (JVM), а `java.exe` является ее ядром. Отметим, что в большинстве современных браузеров виртуальная машина Java установлена по умолчанию.

Каталог, в котором размещена система Java SDK, может иметь, например, такой вид (рис. 1.2).

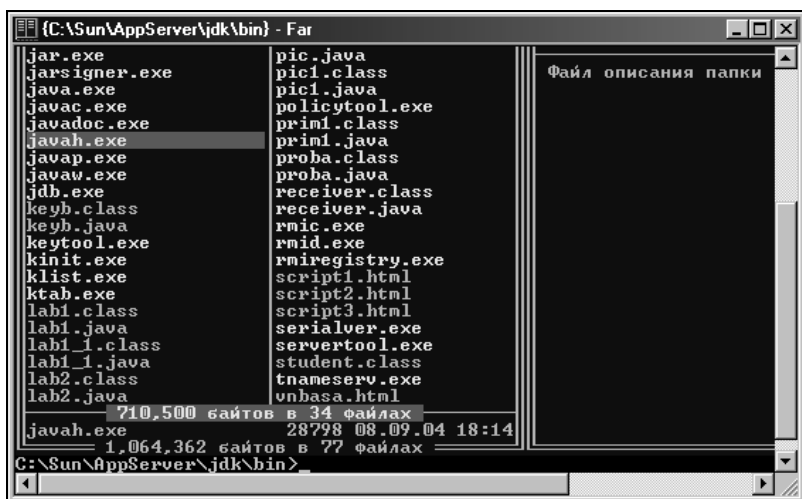


Рис. 1.2. Каталог с размещенной в нем системой Java SDK

В этом практикуме нам понадобятся следующие программы SDK:

- ☐ `javac.exe` — компилятор исходных Java-файлов;
- ☐ `java.exe` — программа выполнения классов;

- ❑ `jar.exe` — программа создания архивов;
- ❑ `packager.exe` — программа создания компонентов Java BEANS.

При первоначальной работе с Java полезны такие утилиты, как `javadoc.exe` — используется для документирования программ, и `appletviewer.exe` — используется для запуска апплетов.

Программирование "без классов"

Программировать без классов уже практически нельзя — это не модно, да и становится крайне тяжело, когда речь идет о сколько-нибудь приемлемом интерфейсе. Имейте в виду, что Java-приложение состоит из классов, где класс — это как функция в языке C или процедура в Delphi. Поэтому заголовок этого раздела не следует понимать буквально, а словосочетание "без классов" взято в кавычки. Разумеется, можно использовать консольные приложения, т. е. приложения без привычных нам окон. Но вот уже для работы с файлами или вывода на консоль вам понадобятся классы. В этом разделе мы намереемся по возможности использовать в программе простые переменные. Например, мы хотели бы использовать операции:

```
x = 2;  
y[0] = x - 1;
```

с переменными `x` и `y` и т. д., и т. п.

Итак, создадим простейшее Java-приложение, в котором запрашивается имя пользователя и выводится приветствие. Увы, для первого знакомства это приложение (листинг 1.1) не очень простое!

Листинг 1.1. Приложение, выполняющее приветствие

```
import java.awt.*;  
import java.io.*;  
public class proba  
{  
    static int b;
```

```
static String sname;
static char[] chararray;
public static void main(String args[])
{
    chararray = new char[20];
    int i=0;
    // Вывод вопроса о вашем имени:
    System.out.println("What is Your name?");
    try
    {
        while((b=System.in.read()) !=13)
        {
            chararray[i++]=(char) b;}
    }
    catch(IOException e)
    {
        System.out.println("The error "+e);
    }
    sname=new String(chararray);
    //Вывод строки приветствия:
    System.out.println("Hello, fellow-programmer, "+sname);
}
}
```

Разберем приведенный в листинге код более подробно. Сначала идет подключение библиотек (библиотечных классов):

```
import java.awt.*;
import java.io.*;
```

В языке C вы подключаете их с помощью инструкции `#include`. В Delphi вы используете команду `use`. Так что здесь нет ничего необычного. Далее идет объявление класса (этого мы не избежали!):

```
public class proba
{
    ...    ...
}
```

Наш класс называется `proba`. Файл, где его следует сохранить, набрав текст программы в каком-нибудь редакторе (например, можно использовать Блокнот или встроенный редактор FAR manager, Norton Commander), также должен называться `proba.java`. Все же это еще не классы в традиционном смысле. Объявляем используемые *переменные*:

```
static int b;  
static String sname;  
static char[] chararray;
```

Во всех языках программирования используются переменные разных типов. У нас это `int`, `String` и `char` (целое число, строка и символ соответственно). Слово `static` нам пришлось использовать, так как мы работаем с классами нетрадиционно, т. е. не создаем объекты этих классов. Итак, запомним, что если мы объявляем переменные класса и не создаем объектов класса, то переменные должны дополняться словом `static`. Наконец, добрались до единственного метода — `main()`:

```
public static void main(String args[])  
{  
    chararray = new char[20];  
    int i=0;  
    System.out.println("What is Your name?");  
    try{  
        while((b=System.in.read()) !=13)  
            {chararray[i++]= (char) b; }  
    }  
    catch(IOException e)  
    {System.out.println("The error"+e);}  
    sname=new String(chararray);  
    System.out.println("Hello, fellow-programmer, "+sname);  
}
```

Метод `main()` всегда объявляется так, как записано (запоминаем). Аргументы этого метода — параметры командной строки,

использованной для запуска приложения. Пока что нам это не потребуется. Затем снова следует два объявления переменных:

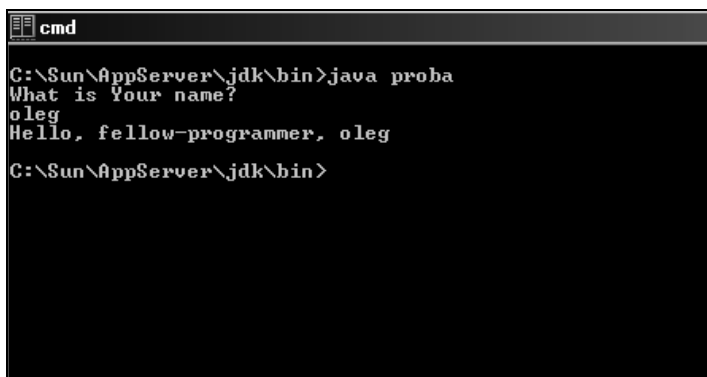
```
// Объявляется массив символов, состоящий из 20 элементов:  
chararray = new char[20];  
int i=0;
```

Где слово *static*? Запомним, что внутри методов это слово использовать нельзя. *Переменные класса* должны объявляться вне методов, но внутри объявлений классов. Внутри методов объявляются локальные переменные методов.

Далее идет команда:

```
System.out.println("What is Your name?");
```

Эта команда, между прочим, использует имя класса `System`. Запомним эту команду для вывода строк на консоль. Результат ее действия мы вскоре увидим. А вот далее в цикле выполняется чтение символов до тех пор, пока не будет прочитан символ `<Enter>` (его код 13). Отсюда начинается то программирование "без классов", о котором мы и намеревались поговорить. Пока не будем "разжевывать" эту программу, но приведем на рис. 1.3 результат ее выполнения.



```
cmd  
C:\Sun\AppServer\jdk\bin>java proba  
What is Your name?  
oleg  
Hello, fellow-programmer, oleg  
C:\Sun\AppServer\jdk\bin>
```

Рис. 1.3. Результат выполнения программы, приведенной в листинге 1.1

Для того чтобы скомпилировать программу, используем командную строку MS-DOS:

```
javac proba.java
```

Программа `javac` — это и есть компилятор Java-программ. Чтобы ввести эту команду, например, из программы Norton Commander, наберите в командной строке `>cmd`.

Если ошибок нет, система создаст файл `proba.class` и разместит его в том же каталоге, где находится файл `proba.java`. Если файл `proba.java` лежит в каталоге, отличном от того, в котором размещен файл `javac.exe`, то следует указать путь к файлу `proba.java`, например:

```
javac c:\temp\proba.java  
(указан путь c:\temp\proba.java).
```

Запускаем класс на выполнение командой

```
java proba
```

(Найдите этот файл на рис. 1.3.)

Заметим, что если `java.exe` находится в каталоге, отличном от того, в котором расположен созданный нами класс `proba.class`, то следует предварительно выполнить команду:

```
set classpath=%classpath%;c:\temp\proba
```

Эту команду следует ввести из командной строки DOS; такую возможность предоставляет также, например, FAR manager или Norton Commander.

Теперь поговорим о программировании. Предполагается, что у вас уже есть кое-какой опыт, но, тем не менее, двигаемся все же с "нуля".

Традиционных команд сравнительно мало:

- ☐ присваивание (=);
- ☐ проверка условия (if);
- ☐ циклы (for и while).

Существует еще команда `goto`, но в данном пособии ее практически не используем (как утверждают специалисты, про нее лучше забыть сразу, потому как в Java это ключевое слово зарезервировано, но самой команды нет).

Традиционное программирование предполагает правильную запись команд для получения задуманного результата. Программы, написанные людьми, не имеющими опыта (даже если это доктора физико-математических наук), зачастую вызывают недоумение.

При написании кода вы работаете с именами переменных. Эти имена должны что-то выражать. Например, пусть требуется ввести какое-нибудь слово и подсчитать его длину (число символов). Обозначим это слово переменной `slovo`, объявим ее и присвоим ей значение пустой строки. *Присваивание* реализуется оператором = следующим образом:

```
static String slovo="";
```

Теперь значение переменной `slovo` равно пустой строке `""`. Если вам не нравится пустая строка, то присвоим переменной `slovo` иное значение. Например:

```
slovo="ур-па".
```

Длина слова возвращается в результате выполнения команды `slovo.length()`.

Поскольку метод `System.in.read()` читает байты (единицы информации), соответствующие символам строки, набираемым на клавиатуре), то объявляем массив байтов и создаем его в памяти:

```
static char[] chararray = new char[20];
```

```
static int i=0;
```

Массив — это множество переменных; имена переменных совпадают с именем массива, но дополняются номерами (индексами). Например, `chararray[0]` — первая переменная массива `chararray`; `chararray[1]` — вторая переменная массива `chararray` и т. д. Создание массива выполняет команда `new`, которая использует базовый тип (в данном случае `char`) и размерность массива (указывается в квадратных скобках).

Организуем цикл для чтения:

```
try{
    while(( int b=System.in.read()) !=13)
    {chararray[i++]=(char) b;} //тело цикла
}
```

Цикл `while` работает так. В скобках записывается условие, необходимое для его выполнения:

```
((int b=System.in.read()) !=13)
```

А именно, указывается, что `b = System.in.read()`, а само проверяемое условие таково: `b != 13` (`b` не равно 13). Если нажимаемая клавиша имеет код, отличный от 13, то символ попадает в массив `chararray` в ячейку `i` и немедленно `i` увеличивается на 1 (это выполняется командой `i++`). Нам надо внимательно следить, чтобы число вводимых символов не превысило 20, так как мы объявили массив именно этого размера. Запомним, что любой цикл имеет тело. *Тело цикла* ограничивается фигурными скобками `{...}`, следующими сразу за условием цикла. То, что в них записывается, выполняется многократно, пока верно условие цикла. Выйти из цикла можно с помощью команды `break`. Итак, внутри нашего цикла считывается очередной символ с клавиатуры по команде:

```
int b=System.in.read();
```

Но значение считанного символа записывается в целую переменную `b`. Конечно, вы знаете, что символы имеют свои числовые коды, представляемые байтами. Например, код пробела — 32. Однако при записи в массив `chararray` числовой код преобразуется в настоящий символ, который данный код определяет. То, что выполняет команда

```
chararray[i++]=(char) b;
```

означает преобразование числа `b` в соответствующий символ `(char)b` (`b` и `(char)b`, по сути, не одно и то же). Это и есть *преобразование типов*. В нашей программе мы еще раз используем преобразование типов при выводе строки на консоль:

```
// строка формируется из символов — представителей другого
// типа:
slovo=new String(chararray);
System.out.println("The word "+slovo+ " was read with
length "+slovo.trim().length());
//выводим строку на консоль
```

Команда `slovo.trim().length()` содержит две последовательно исполняемые операции: `trim()` — удаляет пробелы с начала и конца строки, и `length()` — возвращает длину строки.

Теперь приведем итоговую программу (листинг 1.2), снабженную комментариями. *Комментарии* указываются после двух раздельных наклонных черт ("слэш").

Листинг 1.2. Приложение, выполняющее подсчет длины строки

```
import java.awt.*; // Подключаем библиотеки классов
import java.io.*;

public class proba // Объявляем единственный класс
{
    static int b; // Объявляем типы переменных класса
    static String slovo="";
    // Объявляем массив символов и создаем пустой массив:
    static char[] chararray=new char[20];
    // Определяем единственный метод приложения:
    public static void main(String args[])
    {
        int i=0; // Внутреннее объявление переменной i в методе
                // main()
        System.out.println("Input a word ->"); // Вывод строки
                                                // с подсказкой

        try
        {
            // Конструкция try catch используется для защиты,
            // о чем будет рассказано позже
            while((b=System.in.read()) !=13) // Организуем цикл,
            // try пока не рассматриваем — это к циклу не имеет
            // отношения
            {
                chararray[i++]=(char) b; } // Тело цикла
        }
        catch(IOException e)
        {
            System.out.println("The error"+e);
        }
        // Преобразуем массив chararray символов в строку slovo:
        slovo=new String(chararray);
```