**I/O**

подсистема ввода-вывода Java очень обширна и включает множество классов, интерфейсов и методов. Отчасти это объясняется тем, что в Java определены фактически две полноценные подсистемы ввода-вывода:

**одна — для обмена байтами, другая — для обмена символами.**

В Java операции ввода-вывода реализованы на основе потоков. Поток — это

абстрактная сущность, представляющая устройства ввода-вывода, которая выдает и получает информацию. За связь потоков с физическими устройствами отвечает подсистема ввода-вывода, что позволяет работать с разными устройствами, используя одни и те же классы и методы. Например, **методы вывода на**

**консоль в равной степени могут быть использованы для записи данных в дисковый файл**. Для реализации потоков используется иерархия классов, содержащихся в пакете java.io.

**Говоря проще есть уже написанные классы которые предоставляют функционал для работы с потоками(ввод/вывод). Или ещё проще объекты от этих классов и есть в какой то мере эти самые потоки, а их методы и позволяют работать с вводом/выводом.**

—-----------------------------------**Байтовые и символьные потоки**—-------------------------------------------------

В современных версиях Java определены два типа потоков: **байтовые и символьные**. (Первоначально в Java были доступны только байтовые потоки, но вскоре были реализованы и символьные.) Байтовые потоки предоставляют удобные средства для управления вводом и выводом байтов. Например, их можно использовать для чтения и записи двоичных данных. Потоки этого типа особенно удобны при работе с файлами. С другой стороны, символьные потоки ориентированы на обмен символьными данными. В них применяется кодировка Unicode, и поэтому их легко интернационализировать. Кроме того, в некоторых случаях символьные потоки более эффективны по сравнению с байтовыми потоками.

Необходимость поддерживать два разных типа потоков ввода-вывода привела к созданию двух иерархий классов: одна — для байтовых, другая — для символьных данных. Из-за того что число классов достаточно велико, на первый взгляд подсистема ввода-вывода кажется сложнее, чем она есть на самом деле.

Просто знайте, что в большинстве случаев функциональные возможности символьных потоков идентичны возможностям байтовых.

вместе с тем на самом нижнем уровне все средства ввода-вывода имеют байтовую организацию. Символьные потоки лишь предоставляют удобные и эффективные средства, адаптированные к специфике обработки символов.

—--------------------------------------------**Классы байтовых потоков**—--------------------------------------------------------

Байтовые потоки определены с использованием двух иерархий классов, на вершинах которых находятся абстрактные классы InputStream и OutputStream соответственно.

В классе **InputStream** определены свойства, общие для байтовых потоков ввода, а в классе

**OutputStream** — свойства, общие для байтовых потоков вывода

Производными от классов InputStream и OutputStream являются конкретные подклассы, реализующие различные функциональные возможности и учитывающие особенности обмена данными с разными устройствами, например ввода-вывода в файлы на диске.

**InputStream** Абстрактный класс, описывающий потоковый ввод

**BufferedInputStream** Буферизованный входной поток

**ByteArrayInputStream** Входной поток для чтения из байтового массива

**DataInputStream** Входной поток, включающий методы для чтения стандартных типов данных Java

**FileInputStream -Входной поток для чтения из файла**

**FilterInputStream** Реализация класса InputStream

**ObjectInputStream** Входной поток для объектов

**PipedInputStream** Входной канал

**PushbackInputStream** Входной поток, позволяющий возвращать байты обратно в поток

**SequenceInputStream** Входной поток, сочетающий в себе несколько потоков, которые читаются последовательно, один после другого

**OutputStream** Абстрактный класс, описывающий потоковый вывод

**BufferedOutputStream** Буферизованный выходной поток

**ByteArrayOutputStream** Выходной поток для записи в байтовый массив

**DataOutputStream** Выходной поток, включающий методы для записи стандартных типов данных Java

**FileOutputStream** Выходной поток для записи в файл

**FilterOutputStream** Реализация класса OutputStream

**ObjectOutputStream** Выходной поток для объектов

**PipedOutputStream** Выходной канал

**PrintStream** Выходной поток, включающий методы print() и println()

—---------------------------------------------------**Классы символьных потоков**—----------------------------------------

Символьные потоки также определены с использованием двух иерархий классов, вершины которых на этот раз представлены абстрактными классами Reader и Writer соответственно. Класс Reader и его подклассы используются для чтения, а класс Writer и его подклассы — для записи данных. Конкретные Глава 10. Ввод-вывод данных 385 классы, производные от классов Reader и Writer, оперируют символами в кодировке Unicode. Классы, производные от классов Reader и Writer, предназначены для выполнения различных операций ввода-вывода символов. В целом символьные классы представляют собой аналоги соответствующих классов, предназначенных для работы с байтовыми потоками.

**Reader** Абстрактный класс, описывающий символьный ввод

**BufferedReader** Буферизованный входной символьный поток

**CharArrayReader** Входной поток для чтения из символьного массива

**FileReader Входной поток для чтения из файла**

**FilterReader** Фильтрующий входной поток

**InputStreamReader** Входной поток, транслирующий байты в символы

**PipedReader** Входной канал

**LineNumberReader** Входной поток, подсчитывающий строки

**PushbackReader** Входной поток, позволяющий возвращать символы обратно в поток

**StringReader** Входной поток для чтения из строки

**Writer** Абстрактный класс, описывающий символьный вывод

**BufferedWriter** Буферизованный выходной символьный поток

**CharArrayWriter** Выходной поток для записи в символьный массив

**FileWriter** Выходной поток для записи в файл

**FilterWriter** Фильтрующий выходной поток

**OutputStreamWriter** Выходной поток, транслирующий символы в байты

**PipedWriter** Выходной канал

**PrintWriter** Выходной поток, включающий методы print() и println()

**StringWriter** Выходной поток для записи в строку

во все программы на Java автоматически импортируется

пакет java.lang, в котором определен класс System, инкапсулирующий некоторые свойства среды выполнения. Помимо прочего, в нем содержатся предопределенные переменные **in**, **out** и **err**, представляющие стандартные потоки ввода-вывода. Эти поля объявлены как public, final и static, т.е. к ним

можно обращаться из любой другой части программы, не ссылаясь на конкретный объект типа System.

**Переменная System.out** ссылается на стандартный выходной поток, который по умолчанию связан с консолью.

**Переменная System.in** ссылается на стандартный входной поток, который по умолчанию связан с клавиатурой.

**Переменная System.err** ссылается на стандартный поток ошибок, который, по умолчанию связан с консолью.

При необходимости каждый из этих потоков может быть перенаправлен на любое другое устройство.

**Поток System.in** — это объект типа InputStream(или экземпляр класса), благодаря этому обеспечивается автоматический доступ к методам, определенным в классе InputStream.

System.out и System.err — объекты типа PrintStream. Все эти потоки — байтовые

**поток System.out** — байтовый, он широко используется для побайтового вывода данных на консоль.

Вывод данных на консоль проще всего осуществлять с помощью уже знакомых вам методов print() и

println(). Эти методы определены в классе PrintStream который отвечает за создание выходного потока

—--------------**Чтение и запись файлов с использованием байтовых потоков**—--------------

Язык Java предоставляет множество классов и методов, позволяющих читать и записывать данные из файлов и в файлы. Разумеется, чаще всего приходится обращаться к файлам, хранящимся на дисках. В Java все файлы имеют байтовую организацию, и поэтому для побайтового чтения и записи данных из файла и в файл предусмотрены соответствующие методы. Таким образом, файловые операции с использованием байтовых потоков довольно распространены.

Байтовые потоки, связанные с файлами, создаются с помощью классов **FileInputStream** или **FileOutputStream**. Чтобы открыть файл, достаточно создать объект одного из этих классов, передав конструктору имя файла в качестве параметра. Открытие файла необходимо для того, чтобы с ним можно было выполнять файловые операции чтения и записи. Другими словами объект, из которого (с помощью которого) можно считать данные, называется потоком ввода, а объект, в который (с помощью которого) можно записывать данные, - потоком вывода. Например, если надо считать содержимое файла, то применяется поток ввода, а если надо записать в файл - то поток вывода.

Для наглядного практического примера смотри project UP.Input\_Output