Глава 9

ФАЙЛЫ. ПОТОКИ ВВОДА/ВЫВОДА

Потоки ввода/вывода используются для передачи данных в файловые потоки, на консоль или на сетевые соединения. Потоки представляют собой объекты соответствующих классов. Библиотека ввода/вывода предоставляет пользователю большое число классов и методов и постоянно обновляется.

Класс File

Для работы с физическим файлами и каталогами (директориями), расположенными на внешних носителях, в приложениях Java используются классы из пакета **java.io**.

Класс **File** служит для хранения и обработки в качестве объектов каталогов и имен файлов. Этот класс не содержит методы для работы с содержимым файла, но позволяет манипулировать такими свойствами файла, как права доступа, дата и время создания, путь в иерархии каталогов, создание, удаление файла, изменение его имени и каталога и т.д.

Объект класса **File** создается одним из нижеприведенных способов:

```
File myFile = new File("\\com\\myfile.txt");
File myDir  = new File("c:\\jdk1.6.0\\src\\java\\io");
File myFile = new File(myDir, "File.java");
File myFile = new File("c:\\com", "myfile.txt");
File myFile = new File(new URI("Интернет-адрес"));
```

В первом случае создается объект, соответствующий файлу, во втором – подкаталогу. Третий и четвертый случаи идентичны. Для создания объекта указывается каталог и имя файла. В пятом – создается объект, соответствующий адресу в Интернете.

При создании объекта класса **File** любым из конструкторов компилятор не выполняет проверку на существование физического файла с заданным путем.

Существует разница между разделителями, употребляющимися при записи пути к файлу: для системы Unix – "**/**", а для Windows – "****". Для случаев, когда неизвестно, в какой системе будет выполняться код, предусмотрены специальные поля в классе **File**:

```
public static final String separator;
public static final char separatorChar;
```

С помощью этих полей можно задать путь, универсальный в любой системе:

```
File myFile  = new File(File.separator + "com"
        + File.separator + "myfile.txt" );
```

Также предусмотрен еще один тип разделителей – для директорий:

```
public static final String pathSeparator;
public static final char pathSeparatorChar;
```

К примеру, для ОС Unix значение **pathSeparator=";"**, а для Windows – **pathSeparator=":"**.

В классе **File** объявлено более тридцати методов, наиболее используемые из них рассмотрены в следующем примере:

```
/* пример # 1 : работа с файловой системой: FileTest.java */
package chapt09;
import java.io.*;
import java.util.*;

public class FileTest {
   public static void main(String[] args) {
   //с объектом типа File ассоциируется файл на диске FileTest2.java
       File fp = new File("chapt09" + File.separator
             + "FileTest2.java");
       if(fp.exists()) {
          System.out.println(fp.getName() + " существует");

         if(fp.isFile()) {//если объект – дисковый файл
          System.out.println("Путь к файлу:\t"
                        + fp.getPath());
          System.out.println("Абсолютный путь:\t"
                        + fp.getAbsolutePath());
          System.out.println("Размер файла:\t"
                        + fp.length());
          System.out.println("Последняя модификация :\t"
                        + new Date(fp.lastModified()));
          System.out.println("Файл доступен для чтения:\t"
                        + fp.canRead());
          System.out.println("Файл доступен для записи:\t"
                        + fp.canWrite());
          System.out.println("Файл удален:\t"
                        + fp.delete());
             }
       } else
          System.out.println("файл " + fp.getName()
                        + " не существует");
       try{
          if(fp.createNewFile())
             System.out.println("Файл " + fp.getName()
                        + " создан");
       } catch(IOException e) {
          System.err.println(e);
       }
//в объект типа File помещается каталог\директория
// в корне проекта должен быть создан каталог com.learn с несколькими файлами
    File dir = new File("com" + File.separator + "learn");
          if (dir.exists() && dir.isDirectory())/*если объект
                                        является каталогом и если этот
                                        каталог существует */
```

```
            System.out.println("каталог "
                  + dir.getName() + " существует");
         File[] files = dir.listFiles();
         for(int i = 0; i < files.length; i++){
            Date date = new Date(files[i].lastModified());
            System.out.print("\n" + files[i].getPath()
                  + " \t| " + files[i].length() + "\t| "
                  + date.toString());
          //использовать toLocaleString() или toGMTString()
       }
      // метод listRoots() возвращает доступные корневые каталоги
      File root = File.listRoots()[1];
      System.out.printf("\n%s %,d из %,d свободно.",
root.getPath(),root.getUsableSpace(),root.getTotalSpace());
   }
}
```

В результате файл **FileTest2.java** будет очищен, а на консоль выведено:

```
FileTest2.java существует
Путь к файлу:     chapt09\FileTest2.java
Абсолютный путь:  D:\workspace\chapt09\FileTest2.java
Размер файла:     2091
Последняя модификация : Fri Mar 31 12:26:50 EEST 2006
Файл доступен для чтения:     true
Файл доступен для записи:     true
Файл удален:      true
Файл FileTest2.java создан
каталог learn существует
com\learn\bb.txt  | 9   | Fri Mar 24 15:30:33 EET 2006
com\learn\byte.txt| 8   | Thu Jan 26 12:56:46 EET 2006
com\learn\cat.gif | 670 | Tue Feb 03 00:44:44 EET 2004
C:\ 3 665 334 272 из 15 751 376 896 свободно.
```

У каталога как объекта класса **File** есть дополнительное свойство – просмотр списка имен файлов с помощью методов **list()**, **listFiles()**, **listRoots()**.

Байтовые и символьные потоки ввода/вывода

При создании приложений всегда возникает необходимость прочитать информацию из какого-либо источника и сохранить результат. Действия по чтению/записи информации представляют собой стандартный и простой вид деятельности. Самые первые классы ввода/вывода связаны с передачей и извлечением последовательности байтов.

Потоки ввода последовательности байтов являются подклассами абстрактного класса **InputStream**, потоки вывода – подклассами абстрактного класса **OutputStream**. Эти классы являются суперклассами для ввода массивов байтов, строк, объектов, а также для выбора из файлов и сетевых соединений. При работе с файлами используются подклассы этих классов соответственно