Ввод текста

Если вам необходимо считывать числовые данные из текстового файла класс BufferedReader не подойдет, т.к. в нем отсутствуют методы для чтения числовых данных.

Поэтому для ввода рекомендуется применять не этот класс, а класс Scanner.

Сохранение объектов в текстовом формате

Рассмотрим пример программы, сохраняющей массив записей типа Employee в текстовом файле. Каждая запись сохраняется в отдельной строке. Поля экземпляра отделяются друг от друга разделителем. В качестве разделителя в данном примере используется знак вертикальной черты (|). Другим распространенным разделителем является знак двоеточия (:).

Мы пока что не будем касаться того, что может произойти, если символ | встретится непосредственно в одной из сохраняемых символьных строк. Ниже приведен образец сохраняемых записей:

```
Harry Hacker|35500|1989|10|1
Carl Cracker|75000|1987|12|15
Tony Tester|38000|1990|3|15
```

Процесс записи происходит очень просто. Для этой цели применяется класс PrintWriter, поскольку запись выполняется в текстовый файл. Все поля просто записываются в файл, завершаясь символом |, а если это последнее поле, то комбинацией символов \n. Весь процесс записи совершается в теле приведенного ниже метода writeData(), который описан в классе Employee.

```
public void writeData(PrintWriter out) throws IOException
{
GregorianCalendar calendar = new GregorianCalendar();
calendar.setTime(hireDay);
out.println(name + "|"
+ salary + "|"
+ calendar.get(Calendar.YEAR) + "|"
+ (calendar.get(Calendar.MONTH) + 1) + "|"
+ calendar.get(Calendar.DAY_OF_MONTH));
}
```

Что касается считывания записей, то процесс выполняется построчно с разделением полей. Для чтения каждой строки служит поток сканирования (Scanner), а затем полученная строка разбивается на лексемы с помощью метода String.split(), как показано ниже.

```
public void readData(Scanner in)
{String line = in.nextLine();
String[] tokens = line.split("\\|");
name = tokens[0];
salary = Double.parseDouble(tokens[1]);
int y = Integer.parselnt(tokens[2]);
```

```
int m = Integer.parseInt(tokens[3]);
int d = Integer.parseInt(tokens[4]);
GregorianCalendar calendar = new GregorianCalendar(y, m - 1, d);
hireDay = calendar.getTime();
}
```

В качестве параметра метода split() служит регулярное выражение, описывающее разделитель. Оказывается, что знак вертикальной черты (|) имеет в регулярных выражениях специальное значение, поэтому он должен обязательно экранироваться знаком \, а тот, в свою очередь, еще одним знаком \, в результате чего получается следующее регулярное выражение: "\\|". Весь исходный код данного примера программы представлен в листинге 1.1. В приведенном ниже статическом методе сначала записывается длина массиву, а затем — каждая запись.

```
void writeData(Employee[] e, PrintWriter out)
```

А в следующем статическом методе сначала считывается длина массива, а затем каждая запись:

```
Employee[] readData(BufferedReader in)
```

В приведенном ниже фрагменте кода считываем данные.

```
int n = in.nextInt ();
in.nextLine(); // перейти на следующую строку
Employee[] employees = new Employee[n];
for (int i = 0; i < n; i++)
{
employees[i] = new Employee();
employees[i].readData(in);
}
```

Вызов метода nextInt() приводит к считыванию длины массива, но не завершающего символа новой строки. Этот символ должен обязательно считаться, чтобы в методе readData () можно было перейти к следующей строке вводимых данных при вызове метода nextLine ().

Листинг 1.1. Исходный код TextFileTest. java

```
package textFile;

import java.io.*;
import java.util.*;

/**
 * @version 1.13 2012-05-30
 * @author Cay Horstmann
 */
public class TextFileTest
{
   public static void main(String[] args) throws IOException
```

```
   {
      Employee[] staff = new Employee[3];

      staff[0] = new Employee("Carl Cracker", 75000, 1987, 12, 15);
      staff[1] = new Employee("Harry Hacker", 50000, 1989, 10, 1);
      staff[2] = new Employee("Tony Tester", 40000, 1990, 3, 15);

      // сохранить записи обо всех сотрудниках в файле employee.dat
      try (PrintWriter out = new PrintWriter("employee.dat", "UTF-8"))
      {
         writeData(staff, out);
      }

      // извлечь все записи в новый массив
      try (Scanner in = new Scanner(
            new FileInputStream("employee.dat"), "UTF-8"))
      {
         Employee[] newStaff = readData(in);

         // вывести прочитанные записи о сотрудниках
         for (Employee e : newStaff)
            System.out.println(e);
      }
   }

   /**
    * Записывает данные обо все сотрудниках из массива в поток
    * выводимых данных
    * @param employees массив записей о сотрудниках
    * @param out Поток выводимых данных
    */
   private static void writeData(Employee[] employees, PrintWriter
out) throws IOException
   {
      // записать количество сотрудников
      out.println(employees.length);

      for (Employee e : employees)
         writeEmployee(out, e);
   }

   /**
    * читает записи о сотрудника из scanner в массив
```

```
    * @param in поток сканирования вводимых данных
    * @return массив записей о сотрудниках
    */
   private static Employee[] readData(Scanner in)
   {
      // извлечь размер массива
      int n = in.nextInt();
      in.nextLine(); // обработать символ новой строки

      Employee[] employees = new Employee[n];
      for (int i = 0; i < n; i++)
      {
         employees[i] = readEmployee(in);
      }
      return employees;
   }

   /**
    * записывает данные о сотрудниках в поток записи выводимых данных
    * @param out поток выводимых данных
    */
   public static void writeEmployee(PrintWriter out, Employee e)
   {
      GregorianCalendar calendar = new GregorianCalendar();
      calendar.setTime(e.getHireDay());
      out.println(e.getName() + "|" + e.getSalary() + "|" +
calendar.get(Calendar.YEAR) + "|"
            + (calendar.get(Calendar.MONTH) + 1) + "|" +
calendar.get(Calendar.DAY_OF_MONTH));
   }

   /**
    * считывает данные о сотрудниках из буфера
    * @param in сканированный поток
    */
   public static Employee readEmployee(Scanner in)
   {
      String line = in.nextLine();
      String[] tokens = line.split("\\|");
      String name = tokens[0];
      double salary = Double.parseDouble(tokens[1]);
      int year = Integer.parseInt(tokens[2]);
      int month = Integer.parseInt(tokens[3]);
```

```
      int day = Integer.parseInt(tokens[4]);
      return new Employee(name, salary, year, month, day);
   }
}
```

Листинг 1.2. Исходный код Employee.java

```
package textFile;

import java.util.*;

public class Employee
{
   private String name;
   private double salary;
   private Date hireDay;

   public Employee()
   {
   }

   public Employee(String n, double s, int year, int month, int day)
   {
      name = n;
      salary = s;
      GregorianCalendar calendar =
               new GregorianCalendar(year, month - 1, day);
      hireDay = calendar.getTime();
   }

   public String getName()
   {
      return name;
   }

   public double getSalary()
   {
      return salary;
   }

   public Date getHireDay()
   {
      return hireDay;
```

```
   }

   public void raiseSalary(double byPercent)
   {
      double raise = salary * byPercent / 100;
      salary += raise;
   }

   public String toString()
   {
      return getClass().getName() + "[name=" + name + ",salary=" + salary + ",hireDay=" + hireDay
            + "]";
   }
}
```