1.3 Data Hierarchy

Data items processed by computers form a **data hierarchy** that becomes larger and more complex in structure as we progress from the simplest data items (called “bits”) to richer ones, such as characters and fields. Figure 1.3 illustrates a portion of the data hierarchy.

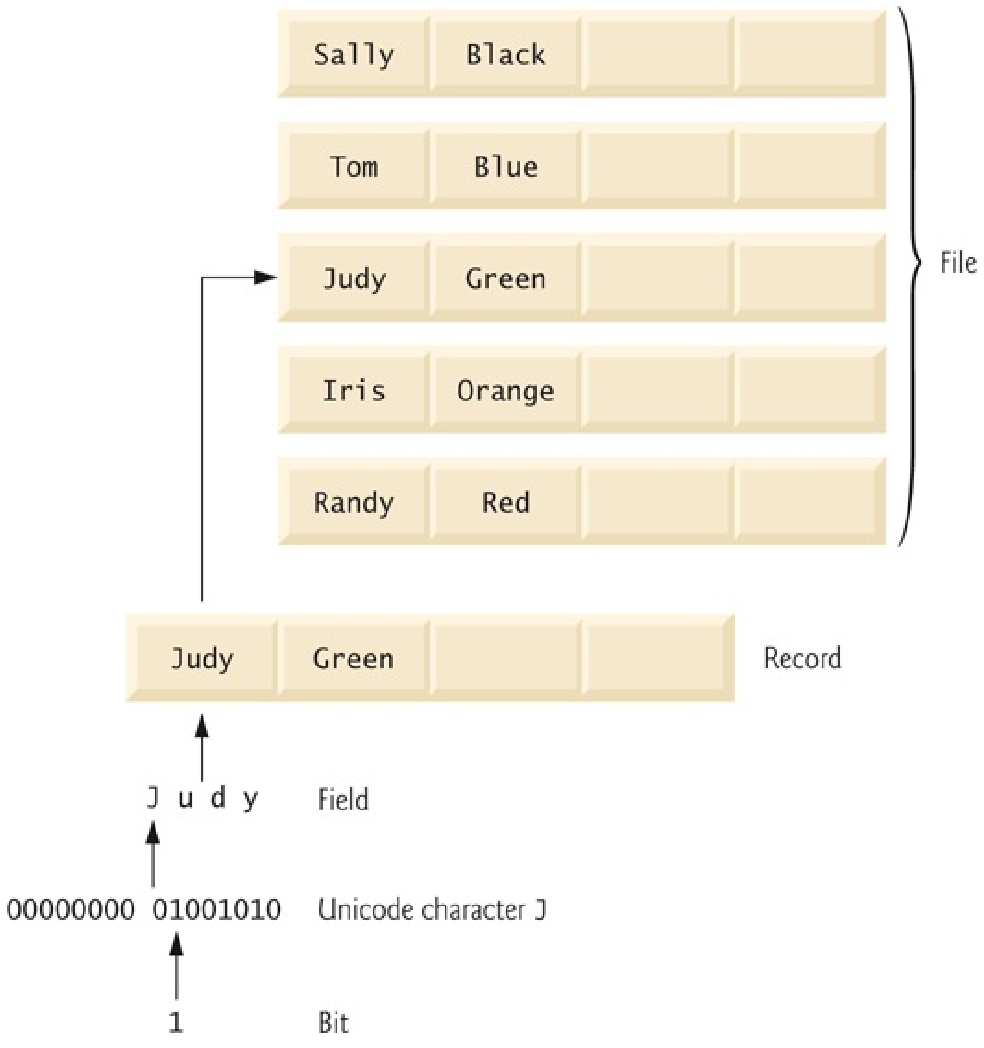

Fig. 1.3

Data hierarchy.

Description

Bits

The smallest data item in a computer can assume the value `0` or the value `1`. It’s called a **bit** (short for “binary digit”—a digit that can assume one of *two* values). Remarkably, the impressive functions performed by computers involve only the simplest manipulations of `0`s and `1`s—*examining a bit’s value, setting a bit’s value* and *reversing a bit’s value* (from `1` to `0` or from `0` to `1`).

Characters

It’s tedious for people to work with data in the low-level form of bits. Instead, they prefer to work with *decimal digits* (0–9), *letters* (A–Z and a–z), and *special symbols* (e.g., $, @, %, &, *, (,), –, +, ", :, ? and /). Digits, letters and special symbols are known as **characters**. The computer’s **character set** is the set of all the characters used to write programs and represent data items. Computers process only `1`s and `0`s, so a computer’s character set represents every character as a pattern of `1`s and `0`s. Java uses **Unicode**® characters that are composed of one, two or four bytes (8, 16 or 32 bits). Unicode contains characters for many of the world’s languages. See Appendix H for more information on Unicode. See Appendix B for more information on the **ASCII (American Standard Code for Information Interchange)** character set—the popular subset of Unicode that represents uppercase and lowercase letters, digits and some common special characters.

Fields

Just as characters are composed of bits, **fields** are composed of characters or bytes. A field is a group of characters or bytes that conveys meaning. For example, a field consisting of uppercase and lowercase letters can be used to represent a person’s name, and a field consisting of decimal digits could represent a person’s age.

Records

Several related fields can be used to compose a **record** (implemented as a `class` in Java). In a payroll system, for example, the record for an employee might consist of the following fields (possible types for these fields are shown in parentheses):

- Employee identification number (a whole number)
- Name (a string of characters)
- Address (a string of characters)
- Hourly pay rate (a number with a decimal point)

- Year-to-date earnings (a number with a decimal point)
- Amount of taxes withheld (a number with a decimal point)

Thus, a record is a group of related fields. In the preceding example, all the fields belong to the *same* employee. A company might have many employees and a payroll record for each.

Files

A **file** is a group of related records. [*Note:* More generally, a file contains arbitrary data in arbitrary formats. In some operating systems, a file is viewed simply as a *sequence of bytes*—any organization of the bytes in a file, such as organizing the data into records, is a view created by the application programmer. You’ll see how to do that in Chapter 15.] It’s not unusual for an organization to have many files, some containing billions, or even trillions, of characters of information.

Database

A **database** is a collection of data organized for easy access and manipulation. The most popular model is the *relational database*, in which data is stored in simple *tables*. A table includes *records* and *fields*. For example, a table of students might include first name, last name, major, year, student ID number and grade point average fields. The data for each student is a record, and the individual pieces of information in each record are the fields. You can *search*, *sort* and otherwise manipulate the data based on its relationship to multiple tables or databases. For example, a university might use data from the student database in combination with data from databases of courses, on-campus housing, meal plans, etc. We discuss databases in Chapter 24, Accessing Databases with JDBC and online Chapter 29, Java Persistence API (JPA).

Big Data

The amount of data being produced worldwide is enormous and growing quickly. According to IBM, approximately 2.5 quintillion bytes (2.5 *exabytes*) of data are created daily,[4] and according to Salesforce.com, as of October 2015 90% of the world's data was created in just the prior 12 months![5] According to an IDC study, the global data supply will reach 40 *zettabytes* (equal to 40 trillion gigabytes) annually by 2020.[6] Figure 1.4 shows some common byte measurements. **Big data** applications deal with massive amounts of data and this field is growing quickly, creating lots of opportunity for software developers. Millions of IT jobs globally already are supporting big data applications.

4. `http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html`

5. `https://www.salesforce.com/blog/2015/10/salesforce-channel-ifttt.html`

6. `http://recode.net/2014/01/10/stuffed-why-data-storage-is-hot-again-really/`

Unit	Bytes	Which is approximately
1 kilobyte (KB)	1024 bytes	10^3 (1024) bytes exactly
1 megabyte (MB)	1024 kilobytes	10^6 (1,000,000) bytes
1 gigabyte (GB)	1024 megabytes	10^9 (1,000,000,000) bytes
1 terabyte (TB)	1024 gigabytes	10^{12} (1,000,000,000,000) bytes
1 petabyte (PB)	1024 terabytes	10^{15} (1,000,000,000,000,000) bytes
1 exabyte (EB)	1024 petabytes	10^{18} (1,000,000,000,000,000,000) bytes
1 zettabyte (ZB)	1024 exabytes	10^{21} (1,000,000,000,000,000,000,000) bytes

Fig. 1.4

Byte measurements.