Third AP Edition

Java Methods

Object-Oriented Programming
and
Data Structures

Maria Litvin
Phillips Academy, Andover, Massachusetts

Gary Litvin
Skylight Software, Inc.

Skylight Publishing
Andover, Massachusetts

Skylight Publishing
9 Bartlet Street, Suite 70
Andover, MA 01810

web: http://www.skylit.com
e-mail: sales@skylit.com
support@skylit.com

Library of Congress Control Number: 2014922396

ISBN   978-0-9824775-6-4

* AP and Advanced Placement are registered trademarks of The College Board, which was not involved in the production of and does not endorse this book.

Appendix D: `EasyReader`, `EasyWriter`, `EasySound`, `EasyDate`

“Easy” classes are intended for novices. They provide a simplified “façade” for more technical Java library solutions for the same tasks. Each of these classes has a simple example of its use in its source file and in the Javadoc documentation.

Feel free to use and distribute these classes in any way you want.

The source code is available in the `JM\EasyClasses` folder and the compiled classes are collected in `JM\EasyClasses\EasyClasses.jar`. The Javadoc documentation is in `JM\EasyClasses\EasyClassesDocs.zip`. Unzip and click on `index.html` in the `docs` folder.

`EasyReader` and `EasyWriter`

We have provided EasyReader and EasyWriter classes to supplement Java’s stream I/O classes. `EasyReader` lets you read numbers, characters, words, and lines of text from the keyboard and from a text file. `EasyWriter` lets you write these data elements into a text file (or append data to an existing file).

`EasyReader` was written before `java.util.Scanner` came into existence in the Java 5.0 release. `EasyReader` is similar to Scanner; it is a little easier to use than `Scanner` for reading keyboard input because it provides a no-args constructor that creates an `EasyReader` for reading from `System.in`. `EasyReader` is easier to use for reading from a text file because it has a constructor, `EasyReader(String pathname)`. (`Scanner` also has a constructor that takes one `String` parameter, but it interprets it as a string to be scanned for input.) `EasyReader` has a method for reading one character from the console or from a file; `Scanner` does not.

`EasyWriter` allows you to create or open a text file for writing and write text into it using the `print`, `println`, and `printf` methods. It eliminates exception handling and complicated constructors that use wrapper classes.

To open the standard input stream for reading keyboard input use

```
     EasyReader kboard = new EasyReader();
```

`kboard` is the name you give to the input stream (can be anything you like).
To open a text file for reading use

```
     EasyReader inputFile = new EasyReader(pathname);
```

`inputFile` is the name you give to the input stream associated with the file (can be anything you like); `pathname` is a `String` that holds the file name or an absolute or relative pathname for the file.

Call the `bad()` method to check the status of the file. It returns `true` if the file is not opened properly or if there is an error or end of file; false otherwise. For example:

```
     EasyReader inputFile = new EasyReader(pathname);
     if (inputFile.bad())
     {
       System.err.println("Cannot open " + pathname);
       System.exit(1);
     }
```

Examples for reading data from the keyboard or a file:

```
     int n = kboard.readInt();            // reads an integer
     double x = kboard.readDouble();      // reads a double

     char ch = inputFile.readChar();      // reads any character,
                                          //   including whitespace
     String word = inputFile.readWord();  // reads a contiguous string of
                                          //   non-whitespace characters

     // Read and process all lines from a file:
     String line;
     while ((line = inputFile.readLine()) != null)
     {
       // process line:
       ...
     }
```

Notes:

1. `readInt`, `readDouble`, `readChar`, and `readWord` methods do not consume the end of the line after reading the last item. Call `readLine` to get rid of the tail of the line (even if only the newline character is left) before calling `readLine` on the next line.

2. `readInt` and `readDouble` methods do not verify that the next token holds a valid number and return 0 or `Double.NaN`, respectively, if it doesn't.

Call `inputFile.close()` to close the file.

To open a text file for writing use

```
     EasyWriter outputFile = new EasyWriter(pathname);
```

or

```
     EasyWriter outputFile = new EasyWriter(pathname, "app");
```

if you want to append data to an existing file. `outputFile` is the name you give to the output stream associated with the file (can be anything you like); `pathname` is a `String` that holds the file name or an absolute or relative pathname for the file.
Be careful:

> **`new EasyWriter(pathname)` wipes out the contents of the file if it already exists.**

Call the `bad()` method, which returns `true` if the attempt to create the file (or to open the file for appending) has failed; `false` otherwise.

Use `print`, `println`, and `printf` methods, the same way as with `System.out`, to write data to a file. For example:

```
     outputFile.print("x = ");
     outputFile.println(x);
```

or

```
     outputFile.printf("x = %5.2f\n", x);
```

Call `outputFile.println()` to write a blank line.
Call `outputFile.close()` to close the file.

Note:

If you forget to close the file, some of the data may remain in the output buffer but not written to the file.

EasySound

This class provides an easy way to load and play a sound clip in a Java program. For example:

```
EasySound bells = new EasySound("bells.wav");
...
bells.play();
```

EasyDate

The `EasyDate` class handles dates in a simple manner. `EasyDate` has a method that adds a number of days to this date, and a method that calculates the number of days from this date to another one. `EasyDate` objects are immutable.

Example:

```
EasyDate today = new EasyDate();
System.out.println("Today is " + today);

EasyDate tomorrow = today.add(1);
EasyDate yesterday = today.add(-1);

int yr = today.getYear();
System.out.println(yr + " is a leap year: true or false? " +
                                 EasyDate.isLeapYear(yr));

EasyDate myBirthday = new EasyDate(bDayMonth, bDayDay, yr);

if (today.equals(myBirthday))
  System.out.println("Today is my birthday");
else if (yesterday.equals(myBirthday))
  System.out.println("My birthday was yesterday");
else
{
  if (myBirthday.compareTo(today) < 0)
    myBirthday = new EasyDate(bDayMonth, bDayDay, yr + 1);
  System.out.println(today.daysTo(myBirthday) +
                 " days are left until my next birthday");
}
```