43. Yes. That is why it is a legitimate argument for `writeObject`.
44. No. Each time the program is run, the file will get longer.
45. Add the following near the start of `main`:

```
ioStream.setLength(0);
```

Programming Projects

PROJECTS INVOLVING ONLY TEXT FILES

VideoNote
Solution to Programming Project 10.1

1. The text files `boynames.txt` and `girlnames.txt`, which are included in the source code for this book text, contain a list of the 1,000 most popular boy and girl names in the United States for the year 2003 as compiled by the Social Security Administration.

   These are blank-delimited files, where the most popular name is listed first, the second most popular name is listed second, and so on, to the 1,000th most popular name, which is listed last. Each line consists of the first name followed by a blank space and then the number of registered births using that name in the year. For example, the `girlnames.txt` file begins with

   Emily 25494

   Emma 22532

   Madison 19986

   This indicates that Emily was the most popular name with 25,494 registered namings, Emma was the second most popular with 22,532, and Madison was the third most popular with 19,986.

   Write a program that reads both the girl and boy files into memory using arrays. Then, allow the user to input a name. The program should search through both arrays. If there is a match, then it should output the popularity ranking and the number of namings. The program should also indicate if there is no match.

   For example, if the user enters the name "Justice," then the program should output

```
Justice is ranked 456 in popularity among girls with 655 namings.
Justice is ranked 401 in popularity among boys with 653 namings.
```

   If the user enters the name "Walter," then the program should output

```
Walter is not ranked among the top 1000 girl names.
Walter is ranked 356 in popularity among boys with 775 namings.
```

2. Write a program that will count the total occurrences of the number '10' in a text file of strings representing numbers of type `int` and will show the value of the count on the screen once the whole file is read. The file contains the following numbers separated by space.

   10 4 7 8 10 34 11 10 15 6 10

3. Write a program that takes its input from a text file of strings representing numbers of type `double` and outputs the average of the numbers in the file to the screen. The file contains nothing but strings representing numbers of type `double`, one per line.

4. Write a program that takes its input from a text file of strings representing numbers of type `double`. The program outputs to the screen the average and standard deviation of the numbers in the file. The file contains nothing but strings representing numbers of type `double`, one per line. The standard deviation of a list of numbers n_1, n_2, n_3, and so forth is defined as the square root of the average of the following numbers:

   $(n_1 - a)^2$, $(n_2 - a)^2$, $(n_3 - a)^2$, and so forth.

   The number a is the average of the numbers n_1, n_2, n_3, and so forth. *Hint:* Write your program so that it first reads the entire file and computes the average of all the numbers, then closes the file, and then reopens the file and computes the standard deviation. You will find it helpful to first do Programming Project 10.3 and then modify that program in order to obtain the program for this project.

5. Write a program that edits a text file to display each complete sentence with a period at the end in a separate line. Your program should work as follows: Create a temporary file, copy from the source file to a temporary file and perform the required operation. Copy the contents of the temporary file back into the source file. Use a method (or methods) in the class `File` to remove the temporary file. You will also want to use the class `File` for other things in your program. The temporary file should have a name that is different from all existing files so that the existing files are not affected (except for the file being edited). Your program will ask the user for the name of the file to be edited. However, it will not ask the user for the name of the temporary file, but will instead generate the name within the program. You can generate the name any way that is clear and efficient. One possible way to generate the temporary file is to start with an unlikely name, such as `"Temp1"`, and to append a digit, such as `'1'`, until a name is found that does not name an existing file.

6. Write a program that gives and takes advice on program writing. The program starts by writing a piece of advice to the screen and asking the user to type in a different piece of advice. The program then ends. The next person to run the program receives the advice given by the person who last ran the program. The advice is kept in a text file and the content of the file changes after each run of the program. You can use your editor to enter the initial piece of advice in the file so that the first person who runs the program receives some advice. Allow the user to type in advice of any length so that it can be any number of lines long. The user is told to end his or her advice by pressing the Return key two times. Your program can then test to see that it has reached the end of the input by checking to see when it reads two consecutive occurrences of the character `'\n'`.

7. Write a class that keeps track of the top five high scores that could be used for a video game. Internally, the class should store the top scores in a data structure of your choice (the most straightforward way is to use arrays). Each entry consists of a name and a score. The data stored in memory should be synchronized with a text file for persistent storage. For example, here are the contents of a sample file where Ronaldo has the highest score and Pele has the third highest score:

```
Ronaldo
10400
Didier
9800
Pele
9750
Kaka
8400
Cristiano
8000
```

The constructor should test if the file exists. If it does not exist, then the file should be created with blank names for each of the players and a score of 0. If the file does exist, then the data from the file should be read into the class's instance variables. Along with appropriate constructors, accessors, and mutators, add the following methods:

- `void playerScore(String name, int score)`: Whenever a game is over, this method is called with the player's name and final score. If the name is one of the top five, then it should be added to the list and the lowest score should be dropped out. If the score is not in the top five, then nothing happens.
- `String[] getTopNames()`: Returns an array of the names of the top players, with the top player first, the second best player second, etc.
- `int[] getTopScores()`: Returns an array of the scores of the top players, with the highest score first, the second highest score second, etc.

Test your program with several calls to `playerScore` and print out the list of top names and scores to ensure that the correct values are stored. When the program is restarted, it should remember the top scores from the last session.

8. Create a file `WordBuff.txt` that contains the following list of words: MADAM, DAD, RISK, JAVA, MALAYALAM, RACECAR, RADAR, ROTOR, REFER, SEDES, SOLOS, COURSE, STATS, TOROT, TENET, MACHINE, VIRTUAL, STUDENT, PULLUP, PROGRAMME, and CORE. Write a program that reads each word from the file and outputs the number of palindromes in the file.

PROJECTS INVOLVING BINARY FILES

VideoNote
Solution to Programming Project 10.9

9. Write a program that will search a binary file of numbers of type `int` and write the largest and the smallest numbers to the screen. The file contains nothing but numbers of type `int` written to the file with `writeInt`.
10. Write a program that reads grades of type `double` of eight students that the user provides. The grades lie between 0 and 10. These grades should be written to a binary file and read from it. The program outputs the highest and lowest grades achieved by students on the screen. The file contains nothing but numbers of type `double` written to the file with `writeDouble`.
11. Write a program that takes its input from a binary file of numbers of type `double`. The file contains nothing but numbers of type `double` written to the file with

writeDouble. The program outputs to the screen the average and standard deviation of the numbers in the file. The standard deviation of a list of numbers n_1, n_2, n_3, and so forth is defined as the square root of the average of the following numbers:

$(n_1 - a)^2$, $(n_2 - a)^2$, $(n_3 - a)^2$, and so forth.

The number a is the average of the numbers n_1, n_2, n_3, and so forth. *Hint:* Write your program so that it first reads the entire file and computes the average of all the numbers, then closes the file, and then reopens the file and computes the standard deviation. You will find it helpful to first do Programming Project 10.8 and then modify that program in order to obtain the program for this project.

12. Change the definition of the class Person in Display 5.19 to be serializable. Note that this requires that you also change the class Date. Then write a program to maintain a binary file of records of people (records of type Person). Allow commands to delete a record specified by the person's name, to add a record, to retrieve and display a record, and to obtain all records of people within a specified age range. To obtain the age of a person, you need the current date. Your program will ask the user for the current date when the program begins. You can do this with random access files, but do not use random access files for this exercise. Use a file or files that record records with the method writeObject of the class ObjectOutputStream.

13. Programming Projects 6.12 and 6.13 asked you to write a program to play a simple trivia game consisting of five questions. The questions, answers, and point values were hardcoded into array(s). This programming project involves moving the trivia questions into one or more binary files instead, and then loading the trivia questions into memory when the program starts.

    First, write a program that allows an administrator to manage the questions for the trivia game. When the program is run, it should check to see if a data file exists. If the data file exists, then the trivia questions should be loaded from the data file into array(s) in memory. If the data file does not exist, start the program with no trivia questions in memory. The program should then present a menu that allows the administrator to list all trivia items (question, answer, and value) in the database, add a new trivia item, or delete an existing trivia item. Upon exiting the program, the trivia data in memory should be stored to one or more binary files using the writeObject method.

    Second, modify either solution to Programming Project 6.12 or 6.13 to read in the trivia data from the binary file created by the administrator's program. Note that the game is no longer limited to five questions, since an arbitrary number of trivia items may be created by the administrator's program and stored in the binary file(s).