# Labs

## Lab 5.1: Reading/Writing Data

### What is the purpose?

In this lab exercise, you will write a program to create a file namedata.txt if it does not exist. Write 100 integers created randomly into the file using text I/O. Integers are separated by spaces in the file. Read the data back from the file and display the sorted data.

### What are the steps?

- Task 1:

  **Procedure**

  1. Create a Java class and name the java file with .java extension.
  2. Import java.io package to use the File class.
  3. Import java.util package to use the Scanner, Formatter, and Arrays classes.
  4. Create a new text file by the File class.
  5. Construct a new formatter by the Formatter class.
  6. Generate 100 random numbers by Math.random().
  7. Construct a scanner to read data from the text file.
  8. Use Arrays.sort() method to reorder all numbers in ascending order.
  9. Express the result to the console.
  10. Compile the java file using the javac command.
  11. Execute the java class using the java command.
  12. Save screen shots of the output similar to Figures 5-1-1 and 5-1-2.

```
0 0 3 4 4 5 5 6 7 7 11 13 15 16 16 17 18 20 20 20 22 23 25 26 28 33 34 35 37 39 41 41 42
43 43 44 44 44 45 46 47 47 49 50 51 51 51 53 55 55 57 58 58 59 59 61 61 62 62 62 64 67
67 67 68 68 69 70 70 70 72 72 73 74 75 77 77 78 82 83 83 85 85 85 86 86 87 88 88 89 89
90 91 92 95 96 96 98 99 99
```

**Figure 5-1-1**

```
0 0 0 1 2 2 4 7 7 7 11 13 15 16 16 17 18 18 19 19 20 20 21 21 22 27 29 29 30 30 31 31 35
36 36 37 39 39 40 41 45 46 46 47 48 48 49 50 50 52 52 52 53 55 55 57 58 58 60 62 62 66
66 68 70 70 73 77 78 78 79 79 80 80 81 81 83 83 84 85 86 86 87 87 87 88 89 90 90 91 94
94 95 95 96 96 98 98 98 99
```

**Figure 5-1-2**

## Did it work?

- Were you able to create 100 random integers into a file called namedata.txt?
- Were you able to sort the 100 integers in ascending order in the console?

## Lab 5.2: Replaying Text

### What is the purpose?

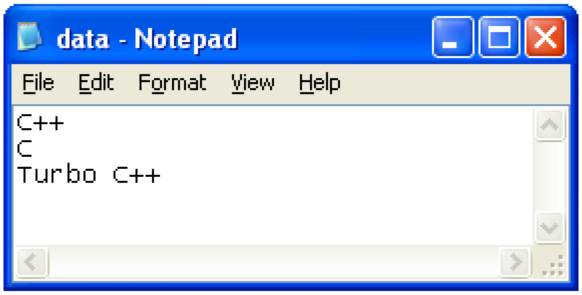
Recall the program ReplaceText.java given in Listing 8.8 on pages 289-290 of your book, which gives a program that replaces text in a source file and saves the change into a new file. Revise this program to save the change into the original file. For example, invoke the commands “java ReplaceText file oldString newString” to replace oldString in the source file with newString.

### What are the steps?

- Task 1:

  **Procedure**

  1. Modify the ReplaceText.java file given on page 289 by replacing the PrintWriter with a StringBuilder so that all the data read from the Scanner class can be save mand.d to a string builder.
  2. Construct a new formatter to format and write back to the source file.
  3. Express the result to the console.
  4. Compile the java file by using the javac command.
  5. Execute the java class by using the java com
  6. Display a sample of the output by following Steps 7-9.
  7. Create a text file called data.txt and type in the data shown in Figure 5-2-1.



```
C++
C
Turbo C++
```

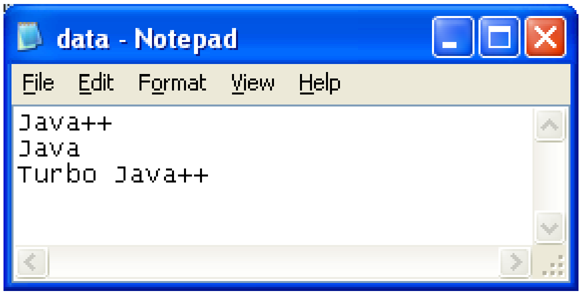
**Figure 5-2-1**

8. Run the ReplaceText program shown in Figure 5-2-2 to replace all the occurrences of “C” with “Java”:

```
java ReplaceText data.txt C Java
```

**Figure 5-2-2**

9. Open the data.txt and check the result as shown in Figure 5-2-3.



```
Java++
Java
Turbo Java++
```

**Figure 5-2-3**

10. Save screenshots similar to Figures 5-2-1 and 5-2-3 and submit them to your instructor.

## Did it work?

Were you able to—

- Revise the ReplaceText class so it can replace text in a text file?
- Implement the command “java ReplaceText file_name oldString newString?

©ITT Educational Services, Inc.