Quiz

Answer the quiz for this chapter online at www.cs.armstrong.edu/liang/intro10e/quiz.html.

MyProgrammingLab™

Programming Exercises

Sections 12.2–12.9

***12.1** (*NumberFormatException*) Listing 7.9, Calculator.java, is a simple command-line calculator. Note that the program terminates if any operand is nonnumeric. Write a program with an exception handler that deals with nonnumeric operands; then write another program without using an exception handler to achieve the same objective. Your program should display a message that informs the user of the wrong operand type before exiting (see Figure 12.12).

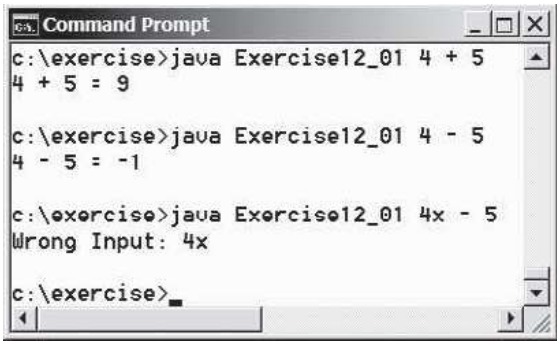

Figure 12.12 The program performs arithmetic operations and detects input errors.

***12.2** (*InputMismatchException*) Write a program that prompts the user to read two integers and displays their sum. Your program should prompt the user to read the number again if the input is incorrect.

***12.3** (*ArrayIndexOutOfBoundsException*) Write a program that meets the following requirements:

- Creates an array with 100 randomly chosen integers.
- Prompts the user to enter the index of the array, then displays the corresponding element value. If the specified index is out of bounds, display the message **Out of Bounds**.

***12.4** (*IllegalArgumentException*) Modify the `Loan` class in Listing 10.2 to throw `IllegalArgumentException` if the loan amount, interest rate, or number of years is less than or equal to zero.

***12.5** (*IllegalTriangleException*) Programming Exercise 11.1 defined the `Triangle` class with three sides. In a triangle, the sum of any two sides is greater than the other side. The `Triangle` class must adhere to this rule. Create the `IllegalTriangleException` class, and modify the constructor of the `Triangle` class to throw an `IllegalTriangleException` object if a triangle is created with sides that violate the rule, as follows:

```
/** Construct a triangle with the specified sides */
public Triangle(double side1, double side2, double side3)
  throws IllegalTriangleException {
  // Implement it
}
```

***12.6** (*NumberFormatException*) Listing 6.8 implements the `hex2Dec(String hexString)` method, which converts a hex string into a decimal number. Implement the `hex2Dec` method to throw a `NumberFormatException` if the string is not a hex string.

***12.7** (*NumberFormatException*) Write the `bin2Dec(String binaryString)` method to convert a binary string into a decimal number. Implement the `bin2Dec` method to throw a `NumberFormatException` if the string is not a binary string.

***12.8** (*HexFormatException*) Exercise 12.6 implements the `hex2Dec` method to throw a `NumberFormatException` if the string is not a hex string. Define a custom exception called `HexFormatException`. Implement the `hex2Dec` method to throw a `HexFormatException` if the string is not a hex string.


VideoNote
HexFormatException

***12.9** (*BinaryFormatException*) Exercise 12.7 implements the `bin2Dec` method to throw a `BinaryFormatException` if the string is not a binary string. Define a custom exception called `BinaryFormatException`. Implement the `bin2Dec` method to throw a `BinaryFormatException` if the string is not a binary string.

***12.10** (*OutOfMemoryError*) Write a program that causes the JVM to throw an `OutOfMemoryError` and catches and handles this error.

Sections 12.10–12.12

****12.11** (*Remove text*) Write a program that removes all the occurrences of a specified string from a text file. For example, invoking

```
java Exercise12_11 John filename
```

removes the string `John` from the specified file. Your program should get the arguments from the command line.

****12.12** (*Reformat Java source code*) Write a program that converts the Java source code from the next-line brace style to the end-of-line brace style. For example, the following Java source in (a) uses the next-line brace style. Your program converts it to the end-of-line brace style in (b).

```
public class Test
{
  public static void main(String[] args)
  {
    // Some statements
  }
}
```

(a) Next-line brace style

```
public class Test {
  public static void main(String[] args) {
    // Some statements
  }
}
```

(b) End-of-line brace style

Your program can be invoked from the command line with the Java source-code file as the argument. It converts the Java source code to a new format. For example, the following command converts the Java source-code file **Test.java** to the end-of-line brace style.

```
java Exercise12_12 Test.java
```

***12.13** (*Count characters, words, and lines in a file*) Write a program that will count the number of characters, words, and lines in a file. Words are separated by whitespace characters. The file name should be passed as a command-line argument, as shown in Figure 12.13.

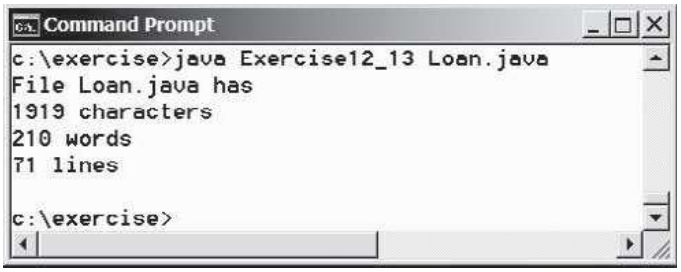

FIGURE 12.13 The program displays the number of characters, words, and lines in the given file.

***12.14** (*Process scores in a text file*) Suppose that a text file contains an unspecified number of scores separated by blanks. Write a program that prompts the user to enter the file, reads the scores from the file, and displays their total and average.

***12.15** (*Write/read data*) Write a program to create a file named **Exercise12_15.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 data in increasing order.

****12.16** (*Replace text*) Listing 12.16, ReplaceText.java, gives a program that replaces text in a source file and saves the change into a new file. Revise the program to save the change into the original file. For example, invoking

```
java Exercise12_16 file oldString newString
```

replaces `oldString` in the source file with `newString`.

*****12.17** (*Game: hangman*) Rewrite Programming Exercise 7.35. The program reads the words stored in a text file named **hangman.txt**. Words are delimited by spaces.

****12.18** (*Add package statement*) Suppose you have Java source files under the directories `chapter1`, `chapter2`, . . . , `chapter34`. Write a program to insert the statement `package chapteri;` as the first line for each Java source file under the directory `chapteri`. Suppose `chapter1`, `chapter2`, . . . , `chapter34` are under the root directory `srcRootDirectory`. The root directory and `chapteri` directory may contain other folders and files. Use the following command to run the program:

```
java Exercise12_18 srcRootDirectory
```

***12.19** (*Count words*) Write a program that counts the number of words in President Abraham Lincoln's Gettysburg address from http://cs.armstrong.edu/liang/data/Lincoln.txt.

****12.20** (*Remove package statement*) Suppose you have Java source files under the directories `chapter1`, `chapter2`, . . . , `chapter34`. Write a program to remove the statement `package chapteri;` in the first line for each Java source file under the directory `chapteri`. Suppose `chapter1`, `chapter2`, . . . , `chapter34` are under the root directory `srcRootDirectory`. The root directory and `chapteri` directory may contain other folders and files. Use the following command to run the program:

```
java Exercise12_20 srcRootDirectory
```

***12.21** (*Data sorted?*) Write a program that reads the strings from file **SortedStrings.txt** and reports whether the strings in the files are stored in increasing order.

If the strings are not sorted in the file, displays the first two strings that are out of the order.

**12.22 (*Replace text*) Revise Programming Exercise 12.16 to replace a string in a file with a new string for all files in the specified directory using the command:

```
java Exercise12_22 dir oldString newString
```

**12.23 (*Process scores in a text file on the Web*) Suppose that the text file on the Web http://cs.armstrong.edu/liang/data/Scores.txt contains an unspecified number of scores. Write a program that reads the scores from the file and displays their total and average. Scores are separated by blanks.

*12.24 (*Create large dataset*) Create a data file with 1,000 lines. Each line in the file consists of a faculty member's first name, last name, rank, and salary. The faculty member's first name and last name for the *i*th line are FirstName*i* and LastName*i*. The rank is randomly generated as assistant, associate, and full. The salary is randomly generated as a number with two digits after the decimal point. The salary for an assistant professor should be in the range from 50,000 to 80,000, for associate professor from 60,000 to 110,000, and for full professor from 75,000 to 130,000. Save the file in **Salary.txt**. Here are some sample data:

FirstName1 LastName1 assistant 60055.95

FirstName2 LastName2 associate 81112.45

. . .

FirstName1000 LastName1000 full 92255.21

*12.25 (*Process large dataset*) A university posts its employees' salaries at http://cs.armstrong.edu/liang/data/Salary.txt. Each line in the file consists of a faculty member's first name, last name, rank, and salary (see Programming Exercise 12.24). Write a program to display the total salary for assistant professors, associate professors, full professors, and all faculty, respectively, and display the average salary for assistant professors, associate professors, full professors, and all faculty, respectively.

**12.26 (*Create a directory*) Write a program that prompts the user to enter a directory name and creates a directory using the `File`'s `mkdirs` method. The program displays the message "Directory created successfully" if a directory is created or "Directory already exists" if the directory already exists.

12.27 (*Replace words*) Suppose you have a lot of files in a directory that contain words **Exercise*i_j*, where *i* and *j* are digits. Write a program that pads a 0 before *i* if *i* is a single digit and 0 before *j* if *j* is a single digit. For example, the word **Exercise2_1** in a file will be replaced by **Exercise02_01**. In Java, when you pass the symbol * from the command line, it refers to all files in the directory (see Supplement III.V). Use the following command to run your program.

```
java Exercise12_27 *
```

12.28 (*Rename files*) Suppose you have a lot of files in a directory named **Exercise*i_j*, where *i* and *j* are digits. Write a program that pads a 0 before *i* if *i* is a single digit. For example, a file named **Exercise2_1** in a directory will be renamed to **Exercise02_1**. In Java, when you pass the symbol * from the command line, it refers to all files in the directory (see Supplement III.V). Use the following command to run your program.

```
java Exercise12_28 *
```

12.29 (*Rename files*) Suppose you have a lot of files in a directory named **Exercise*i_j*, where *i* and *j* are digits. Write a program that pads a 0 before *j* if *j* is a single digit. For example, a file named **Exercise2_1** in a directory will be renamed to **Exercise2_01**. In Java, when you pass the symbol * from the command line, it refers to all files in the directory (see Supplement III.V). Use the following command to run your program.

```
java Exercise12_29 *
```

**12.30 (*Occurrences of each letter*) Write a program that prompts the user to enter a file name and displays the occurrences of each letter in the file. Letters are case-insensitive. Here is a sample run:

```
Enter a filename: Lincoln.txt Enter
Number of A's: 56
Number of B's: 134
...
Number of Z's: 9
```

*12.31 (*Baby name popularity ranking*) The popularity ranking of baby names from years 2001 to 2010 is downloaded from www.ssa.gov/oact/babynames and stored in files named **babynameranking2001.txt**, **babynameranking2002.txt**, . . . , **babynameranking2010.txt**. Each file contains one thousand lines. Each line contains a ranking, a boy's name, number for the boy's name, a girl's name, and number for the girl's name. For example, the first two lines in the file **babynameranking2010.txt** are as follows:

```
1       Jacob     21,875      Isabella      22,731
2       Ethan     17,866      Sophia        20,477
```

So, the boy's name Jacob and girl's name Isabella are ranked #1 and the boy's name Ethan and girl's name Sophia are ranked #2. 21,875 boys are named Jacob and 22,731 girls are named Isabella. Write a program that prompts the user to enter the year, gender, and followed by a name, and displays the ranking of the name for the year. Here is a sample run:

```
Enter the year: 2010 Enter
Enter the gender: M Enter
Enter the name: Javier Enter
Javier is ranked #190 in year 2010
```

```
Enter the year: 2010 Enter
Enter the gender: F Enter
Enter the name: ABC Enter
The name ABC is not ranked in year 2010
```

**12.32 (*Ranking summary*) Write a program that uses the files described in Programming Exercise 12.31 and displays a ranking summary table for the first five girl's and boy's names as follows:

Year	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
2010	Isabella	Sophia	Emma	Olivia	Ava	Jacob	Ethan	Michael	Jayden	William
2009	Isabella	Emma	Olivia	Sophia	Ava	Jacob	Ethan	Michael	Alexander	William
...										
2001	Emily	Madison	Hannah	Ashley	Alexis	Jacob	Michael	Matthew	Joshua	Christopher

**12.33 (*Search Web*) Modify Listing 12.18 WebCrawler.java to search for the word `Computer Programming` starting from the URL http://cs.armstrong.edu/liang. Your program terminates once the word is found. Display the URL for the page that contains the word.