Java Software Solutions 10th edition Lewis Solutions Manual

SKU: 9780137620166-SOLUTIONS

Chapter 1 Exercise Solutions

EX 1.1. Describe the hardware components of your personal computer or of a computer in a lab to which you have access. Include the processor type and speed, storage capacities of main and secondary memory, and types of I/O devices. Explain how you determined your answers.

One possible description:

Processor: Intel Celeron D, 2.53 GHz

Main Memory: 2 GB SDRAM

Secondary Memory: 500 GB Ultra ATA Hard Drive

Peripherals: CD/DVD +/- RW, keyboard, mouse, 17" flat screen monitor, HP LaserJet printer

To find the processor and memory information, I accessed the System Information from the control panel. To find the I/O devices, I looked in the device manager.

EX 1.2. Why do we use the binary number system to store information on a computer?

Devices that store and move information are less expensive and more reliable if they have to represent only one of two possible values or states.

EX 1.3. How many unique items can be represented with each of the following?

a. 1 bit

2 items

b. 3 bits

8 items

c. 6 bits

64 items

d. 8 bits

256 items

e. 10 bits

1024 items

f. 16 bits

16,348 items

EX 1.4. If a picture is made up of 128 possible colors, how many bits would be needed to store each pixel of the picture? Why?

Seven bits are needed to represent each pixel if each pixel can have up to 128 possible colors. This is because there are 128 distinct permutations of 7 bits.

EX 1.5. If a language uses 240 unique letters and symbols, how many bits would be needed to store each character of a document? Why?

Eight bits are needed to store each character of a document written in a language of 240 unique characters and symbols. Seven bits would be sufficient if there were only 128 different characters to represent. Eight bits is sufficient for 256 different characters.

Because 240 is greater than 128, but not greater than 256, at least 8 bits are needed if all characters are represented by the same number of bits.

EX 1.6. How many bits are there in each of the following? How many bytes are there in each?

a. 12 KB

12 KB = 12 x 1024 bytes = 12,288 bytes = 98,304 bits

b. 5 MB

5 MB = 5 x 1,048,576 bytes = 5,242,880 bytes = 41,943,040 bits

c. 3 GB

3 GB = 3 x 1,703,741,824 bytes = 5,111,225,472 bytes = 40,889,803,776 bits

d. 2 TB

2 TB = 2 x 1,099,511,627,776 bytes = 2,199,023,255,552 bytes = approximately 1.76 x 10^{13} bits

EX 1.7. Explain the difference between random-access memory (RAM) and read-only memory (ROM).

Both RAM and ROM are random access devices. RAM (Random Access Memory) can be written to and read from, but ROM (Read-Only Memory) can only be read from.

EX 1.8. A disk is a random-access device but it is not RAM (random-access memory). Explain.

The data on both can be accessed diretly (without reading intervening data). But RAM typically refers to a set of chips that make up main memory, whereas a disk is considered secondary memory. RAM is volatile, and a disk is not.

EX 1.9. **Determine how your computer, or a computer in a lab to which you have access, is connected to others across a network. Is it linked to the Internet? Draw a diagram to show the basic connections in your environment.**

The computers in our lab are connected to a local area network, which is connected to the Internet. (diagram not provided)

EX 1.10. Explain the differences between a local-area network (LAN) and a wide-area network (WAN). What is the relationship between them?

A LAN is designed to span a short distance and to connect a relatively small number of computers. A WAN connects two or more LANs, typically across longer distances such as throughout a group of buildings.

EX 1.11. What is the total number of communication lines needed for a fully connected point-to-point network of eight computers? Nine computers? Ten computers? What is a general formula for determining this result?

Eight computers: 28 communication lines

Nine computers: 36 communication lines

Ten computers: 45 communication lines

General formula for n computers: n(n-1)/2, which represents the sum of the numbers between 1 and n-1.

EX 1.12. Explain the difference between the Internet and the World Wide Web.

The Internet is a network of networks. The World Wide Web is based on a set of software applications that facilitates sharing of information across a network.

EX 1.13. List and explain the parts of the URLs for:

a. your school

http://www.byu.edu, where http stands for HyperText Transfer Protocol, which determines the way the browser should communicate; the machine referenced is www, a typical reference to a Web server; the domain is byu.edu where byu stands for Brigham Young University, and edu indicates that it is an educational institution.

b. the Computer Science department of your school

http://www.cs.byu.edu, in which cs refers to the subdomain within the larger byu.edu domain. So in this case the www machine refers to the standard web server designated by the cs department.

c. your instructor's Web page

http://www.cs.byu.edu/rpburton/info.html/, which refers to a specific file, rpburton/info.html, located on the computer science web server to be transferred to the user's browser for viewing.

EX 1.14. Give examples of the two types of Java comments and explain the differences between them.

One kind of comment begins with a double slash (//) and continues to the end of the line. A second kind of comment begins following an initiating slash-asterisk (/) and terminates immediately preceding a terminating asterisk-slash (*/). The second type of comment can span multiple lines.*

EX 1.15. Which of the following are not valid Java identifiers? Why?

a. Factorial

Valid

b. anExtremelyLongIdentifierIfYouAskMe

Valid

c. 2ndLevel

Invalid because it begins with a digit

d. level2

Valid

e. MAX_SIZE

Valid

f. highest$

Valid

g. hook&ladder

Invalid because it contains an ampersand (&)

EX 1.16. Why are the following valid Java identifiers not considered good identifiers?

a. q

The identifier `q` is a meaningless name.

b. totVal

The idetnifier `totalValue` would be more meaningful than the abbreviation.

c. theNextValueInTheList

Unnecessarily lengthy; `nextValue` would serve as well.

EX 1.17. Java is case sensitive. What does that mean?

Uppercase characters are considered to be distinct from lowercase letters. Therefore the identifier `HELLO` is distinct from `Hello` which is distinct from `hello`.

EX 1.18. What is a Java Virtual Machine? Explain its role.

A Java Virtual Machine (JVM) is a software interpreter that executes Java bytecode. Since bytecode is a low-level representation of a program, but not tied to any particular hardware architecture, any computer with a JVM can execute Java code, no matter what machine it was compiled on. That makes Java architecture-neutral, and therefore highly portable.

EX 1.19. What do we mean when we say that the English language is ambiguous? Give two examples of English ambiguity (other than the example used in this chapter) and explain the ambiguity. Why is ambiguity a problem for programming languages?

Something is ambiguous if it has two or more possible meanings. For example, the statement, "Mary is the nicest teaching assistant who has helped me all day long" might mean 1) of all the teaching assistants who have helped me today, Mary is the nicest, or 2) of those teaching assistants who have helped me for an entire day, Mary is the nicest. As another example, the statement, "Bananas help those who help themselves" might mean 1) bananas are good for those who attend to their own welfare or 2) bananas are good for those who eat as many bananas as they please. If a programming language statement could be interpreted in two or more ways, it would be impossible to predict with certainty how it would be interpreted and what result would be produced.

EX 1.20. Categorize each of the following situations as a compile-time error, run-time error, or logical error.

a. multiplying two numbers when you meant to add them

A logical error

b. dividing by zero

A run-time error

c. forgetting a semicolon at the end of a programming statement

A compile-time error

d. spelling a word wrong in the output

A logical error

e. producing inaccurate results

A logical error

f. typing a `{` when you should have typed `(`

A compile-time error

Chapter 1: Introduction
Lab Exercises

Topics	Lab Exercises
Printing strings	Prelab Exercises Poem
Documentation	Comments
Identifiers	Program Names
Syntax errors	Recognizing Syntax Errors Correcting Syntax Errors

Prelab Exercises

Your task is to write a Java program that will print out the following message (including the row of equal marks):

```
Computer Science, Yes!!!!
=========================
```

An outline of the program is below. Complete it as follows:

a. In the documentation at the top, fill in the name of the file the program would be saved in and a brief description of what the program does.
b. Add the code for the main method to do the printing.

```
// ********************************************************
// File Name:
//
// Purpose:
// ********************************************************

public class CSYes
{
    // -------------------------------------------------------------------------
    //  The following main method prints an exciting
    //  message about computer science
    // -------------------------------------------------------------------------




}
```

Poem

Write a Java program that prints the message, “Roses are red”. Your program will be a class definition containing a *main* method—see the *Lincoln* example in Listing 1.1 of the text if you need guidance. Remember the following:

- The name of the class must match the name of the file (but without the .java extension).
- The *main* method must be inside the class definition (between the first { and the last}).
- The statement that prints the message must be inside *main.*

Compile and run your program. When it works correctly, modify it so that it prints the entire poem:

```
Roses are red
Violets are blue
Sugar is sweet
And so are you!
```

Comments

File *Count.java* contains a Java program that counts from 1 to 5 in English, French, and Spanish. Save this file to your directory and compile and run it to see what it does. Then modify it as follows:

1. Use // style comments to add a comment header at the top of the file that includes the name of the program, your name, and a brief description of what the program does, neatly formatted. Include a delimiter line (e.g., all stars) at the beginning and end of the header.
2. Add a comment before each println that indicates what language the next line is in. Experiment with leaving a blank line before each of these comment lines (in the program itself, not the output). Is the program easier to read with or without these blank lines?
3. Remove one of the slashes from one of your comment lines and recompile the program, so one of the comments starts with a single /. What error do you get? Put the slash back in.
4. Try putting a comment within a comment, so that a // appears after the initial // on a comment line. Does this cause problems?
5. Consult the documentation guidelines in Appendix F of the text. Have you violated any of them? List two things that you could imagine yourself or someone else doing in commenting this program that these guidelines discourage.

Count.java

```
public class Count
{

   public static void main (String[] args)
   {

     System.out.println ("one two three four five");
     System.out.println ("un deux trois quatre cinq");
     System.out.println ("uno dos tres cuatro cinco");

   }

}
```

Program Names

File *Simple.java* contains a simple Java program that prints a message. The identifier that represents the name of this program is *Simple,* but we could have chosen a different identifier subject to certain rules and conventions. An identifier may contain any combination of letters, digits, the underscore character, and the dollar sign, but cannot begin with a digit. Furthermore, by convention, identifiers that represent class names (which includes program names) begin with a capital letter. This means that you won't get an error if your program name doesn't start with a capital letter (as long as what it starts with is legal for an identifier), but it's better style if you do. This may seem arbitrary now, but later when you have lots of identifiers in your programs you'll see the benefit. Of course, the program name should always be reasonably descriptive of the program.

Indicate whether each name below is a legal identifier, and if so, whether it is a good choice to name this program. If the answer to either question is no, explain why. Then save Simple.java to your directory and check your answers by modifying it to try each—note that you will have to change the file name each time to match the program name or you will always get an error.

1. simple (Why do you even have to change the name of the file in this case?)
2. SimpleProgram
3. 1 Simple
4. _Simple_
5. *Simple*
6. $123_45
7. Simple!

```
//  ********************************************
//    Simple.java
//
//    Print a simple message about Java.
//
//  ********************************************

public  class  Simple
{

     public static void main (String[] args)
     {

       System.out.println ("Java rocks!!");

     }

}
```

Recognizing Syntax Errors

When you make syntax errors in your program the compiler gives error messages and does not create the bytecode file. It saves time and frustration to learn what some of these messages are and what they mean. Unfortunately, at this stage in the game many of the messages will not be meaningful *except* to let you know where the first error occurred. Your only choice is to carefully study your program to find the error. In the following you will introduce a few typical errors into a simple program and examine the error messages.

1. Type the following program into a file called Hello.java. (This is the traditional first program a computer scientist writes in a new language.)

```
//  ********************************************
//    Hello.java
//
//    Print a Hello, World message.
//  ********************************************

public class Hello
{
     // -------------------------------------------------------
     // main method -- prints the greeting
     // -------------------------------------------------------
     public static void main (String[] args)
     {
           System.out.println ("Hello, World!");
      }
}
```

   Compile and run the program to see what it does. Then make the changes below, answering the questions as you go.

2. **Class name different from file name.** Delete one l (el) from the name of the class (so the first non-comment line is *public class Helo),* save the program, and recompile it. What was the error message?

3. **Misspelling inside string.** Correct the mistake above, then delete one l from the Hello in the message to be printed (inside the quotation marks). Save the program and recompile it. There is no error message—**why not?** Now run the program. What has changed?

4. **No ending quotation mark in a string literal.** Correct the spelling in the string, then delete the ending quotation mark enclosing the string Hello, World!. Save the program and recompile it. What error message(s) do you get?

5. **No beginning quotation mark in a string literal.** Put the ending quotation mark back, then take out the beginning one. Save and recompile. How many errors this time? Lots, even though there is really only one error. **When you get lots of errors always concentrate on finding the first one listed!!** Often fixing that one will fix the rest. After we study variables the error messages that came up this time will make more sense.

6. **No semicolon after a statement.** Fix the last error (put the quotation mark back). Now remove the semicolon at the end of the line that prints the message. Save the program and recompile it. What error message(s) do you get?

Correcting Syntax Errors

File *Problems.java* contains a simple Java program that contains a number of syntax errors. Save the program to your directory, study it and correct as many of the errors as you can find. Then compile the program; if there are still errors, correct them. Some things to remember:

- Java is case sensitive, so, for example, the identifiers *public*, *Public*, and *PUBLIC* are all considered different. For reserved words such as *public* and *void* and previously defined identifiers such as *String* and *System*, you have to get the case right. You will learn the conventions about case soon, but for now you can look at a sample program in the text to see what case should be used where.
- When the compiler lists lots of errors, fix the first one (or few) and then recompile—often the later errors aren't really errors in the program, they just indicate that the compiler is confused from earlier errors.
- Read the error messages carefully, and note what line numbers they refer to. Often the messages are helpful, but even when they aren't, the line numbers usually are.
- When the program compiles cleanly, run it.

```
//  ****************************************
//     Problems.java
//
//     Provide lots of syntax errors for the user to correct.
//
  ****************************************

public  class  problems
{

      public Static main (string[] args)
      {

        System.out.println      ("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
        System.out.println ("This program used to have lots of problems,");
        System.out.println  ("but if it prints this, you  fixed  them all.")
        System.out.println ("                    *** Hurray!  ***");
        System.out.println   ("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");

      }
}
```