Rogers Cadenhead

Sams **Teach Yourself**

Java™

in 24 Hours

SAMS

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Rogers Cadenhead

Sams Teach Yourself Java™ in 24 Hours

Seventh Edition

SAMS 800 East 96th Street, Indianapolis, Indiana, 46240 USA

Sams Teach Yourself Java™ in 24 Hours, Seventh Edition

ISBN-13: 978-0-672-33702-4
ISBN-10: 0-672-33702-9

Library of Congress Control Number: 2014936457

Printed in the United States of America

Second Printing: December 2014

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact international@pearsoned.com.

Acquisitions Editor
Mark Taber

Managing Editor
Sandra Schroeder

Senior Project Editor
Tonya Simpson

Copy Editor
Barbara Hacha

Indexer
WordWise Publishing Services

Proofreader
Chuck Hutchinson

Technical Editor
Boris Minkin

Editorial Assistant
Vanessa Evans

Cover Designer
Mark Shirar

Compositor
Trina Wurst

Table of Contents

About the Author

Rogers Cadenhead is a writer, computer programmer, and web developer who has written more than 20 books on Internet-related topics, including *Sams Teach Yourself Java in 21 Days.* He maintains the Drudge Retort and other websites that receive more than 20 million visits a year. This book's official website is at www.java24hours.com.

Dedication

I began programming as a 13-year-old on a Timex Sinclair 1000, a computer with a 3.25 MHz processor and 2KB of memory that used a TV as a monitor. I'd like to dedicate this book to the person who bought that computer and never complained when I immediately stole it from him—my dad, Roger Cadenhead, Sr. Thanks, Dad! That led me to this.

Acknowledgments

To the folks at Sams and Pearson—especially Mark Taber, Tonya Simpson, Seth Kerney, Barbara Hacha, and Boris Minkin. No author can produce a book like this on his own. Their excellent work will give me plenty to take credit for later.

To my wife, Mary, and my sons, Max, Eli, and Sam.

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

E-mail: feedback@samspublishing.com

Mail: Reader Feedback
Sams Publishing
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.

Introduction

As the author of computer books, I spend a lot of time lurking in the computer section of bookstores, observing the behavior of readers while I'm pretending to read the latest issue of *In Touch Weekly* magazine.

Because of my research, I've learned that if you have picked up this book and turned to the introduction, I only have 13 more seconds before you put it down and head to the coffee bar for a double-tall-decaf-skim-with-two-shots-of-vanilla-hold-the-whip latte.

So I'll keep this brief: Computer programming with Java is easier than it looks. I'm not supposed to tell you that because thousands of programmers have used their Java skills to get high-paying jobs in software development, web application programming, and mobile app creation. The last thing any programmer wants is for the boss to know that anyone with persistence and a little free time can learn this language, the most popular programming language on the planet. By working your way through each of the one-hour tutorials in *Sams Teach Yourself Java in 24 Hours*, you'll be able to learn Java programming quickly.

Anyone can learn how to write computer programs—even if you can't program a DVR. Java is one of the best programming languages to learn because it's a useful, powerful, modern technology that's embraced by programmers around the world.

This book is aimed at nonprogrammers, new programmers who hated learning the subject, and experienced programmers who want to get up to speed swiftly with Java. It uses Java 8, the brand-new version of the language.

Java is an enormously popular programming language because of the things it makes possible. You can create programs that feature a graphical user interface, design software that makes the most of the Internet, connect to web services, create an app that runs on an Android phone or tablet, and more.

This book teaches Java programming from the ground up. It introduces the concepts in English instead of jargon with step-by-step examples of working programs you will create. Spend 24 hours with this book and you'll be writing your own Java programs, confident in your ability to use the language and learn more about it. You also will have skills that are becoming increasingly important—such as network computing, graphical user interface design, and object-oriented programming.

These terms might not mean much to you now. In fact, they're probably the kind of thing that makes programming seem intimidating and difficult. However, if you can use a computer to create a photo album on Facebook, pay your taxes, or work an Excel spreadsheet, you can learn to write computer programs by reading *Sams Teach Yourself Java in 24 Hours*.

NOTE

At this point, if you would rather have coffee than Java, please reshelve this book with the front cover facing outward on an endcap near a lot of the store's foot traffic.

HOUR 2
Writing Your First Program

As you learned during Hour 1, "Becoming a Programmer," a computer program is a set of instructions that tell a computer what to do. These instructions are given to a computer using a programming language.

During this hour, you create your first Java program by entering it into a text editor. When that's done, you save the program, compile it, and test it out. Then you break it on purpose and fix it again, just to show off.

THIS HOUR'S TO-DO LIST:

- Type a Java program into a text editor.
- Organize a program with bracket marks.
- Store information in a variable.
- Display the information stored in a variable.
- Save, compile, and run a program.

What You Need to Write Programs

As explained in Hour 1, to create Java programs, you must have a programming tool that supports the Java Development Kit (JDK) such as the NetBeans integrated development environment (IDE). You need a tool that can compile and run Java programs and a text editor to write those programs.

With most programming languages, computer programs are written by entering text into a text editor (also called a source code editor). Some programming languages come with their own editor. NetBeans includes its own editor for writing Java programs.

Java programs are simple text files without any special formatting, such as centered text or boldface text. The NetBeans source code editor functions like a simple text editor with some extremely useful enhancements for programmers. Text turns different colors as you type to identify different elements of the language. NetBeans also indents lines properly and provides helpful programming documentation inside the editor.

Because Java programs are text files, you can open and edit them with any text editor. You could write a Java program with NetBeans, open it in Windows Notepad and make changes, and open it again later in NetBeans without any problems.

Creating the `Saluton` Program

The first Java program that you create will display a traditional greeting from the world of computer science: "Saluton mondo!"

To prepare for the first programming project in NetBeans, if you haven't already done so, create a new project called Java24 by following these steps:

1. Choose the menu command File, New Project. The New Project dialog opens.
2. Choose the project category `Java` and the project type `Java Application` and then click Next.
3. Enter `Java24` as the project's name. (If you created a project with this name previously, you see the error message "Project folder already exists and is not empty.")
4. Deselect the Create Main Class check box.
5. Click Finish.

The Java24 project is created in its own folder. You can use this project for the Java programs you write as you progress through this book.

Beginning the Program

NetBeans groups related programs together into a project. If you don't have the Java24 project open, here's how to retrieve it:

1. Choose File, Open Project. A file dialog appears.
2. Find and select the `NetBeansProjects` folder (if necessary).
3. Choose `Java24` and click Open Project.

The Java24 project appears in the Projects pane next to a coffee cup icon and a + sign that can be expanded to see the files and folders that the project contains.

To add a new Java program to the currently open project, choose File, New File. The New File Wizard opens, as shown in Figure 2.1.

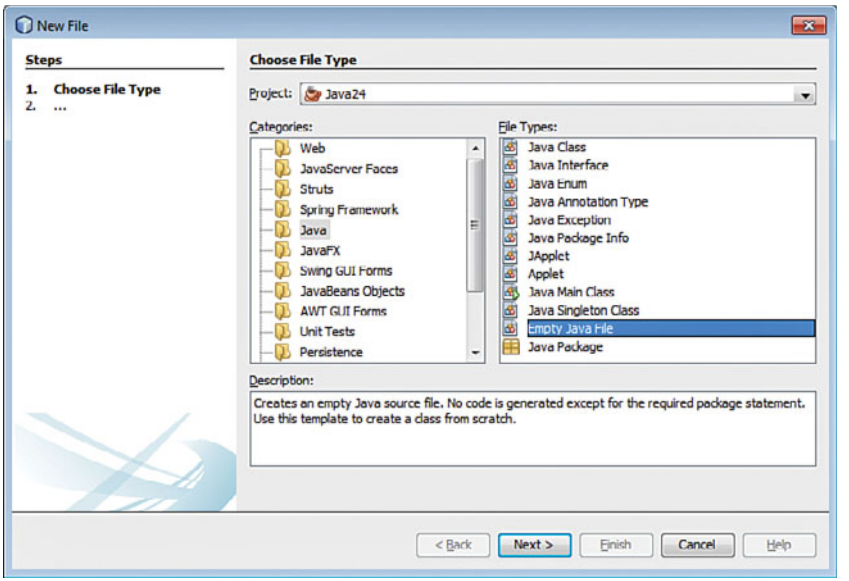

FIGURE 2.1
The New File Wizard.

The Categories pane lists the different kinds of Java programs you can create. Click the Java folder in this pane to see the file types that belong to this category. For this first project, choose the Empty Java File type and click Next.

A New Empty Java File dialog opens. Follow these steps to begin writing the program:

1. In the Class Name field, enter `Saluton`.
2. In the Package field, enter `com.java24hours`.
3. Click Finish.

So you can begin working right away on your program, an empty file named `Saluton.java` opens in the source code editor. Using the editor, begin your Java programming career by entering each line from Listing 2.1. These statements are called the program's source code.

CAUTION

Don't enter the line number and colon at the beginning of each line—these are used in this book to reference specific line numbers.

LISTING 2.1 The `Saluton` Program

```
1: package com.java24hours;
2:
3: class Saluton {
4:     public static void main(String[] arguments) {
5:         // My first Java program goes here
6:     }
7: }
```

Make sure to capitalize everything exactly as shown, and use your spacebar or Tab key to insert the blank spaces in front of Lines 4–6. When you're done, choose File, Save to save the file.

At this point, `Saluton.java` contains the bare-bones form of a Java program. You will create many programs that start exactly like this one, except for the word `Saluton` on Line 3. This word represents the name of your program and changes with each program you write. Line 5 should make sense to you, because it's a sentence in actual English. The rest is probably new to you.

The `class` Statement

The first line of the program is the following:

```
package com.java24hours;
```

A package is a way to group Java programs together. This line tells the computer to make `com.java24hours` the package name of the program.

After a blank line, the third line is this:

```
class Saluton {
```

Translated into English, it means, "Computer, give my Java program the name `Saluton`."

As you might recall from Hour 1, each instruction you give a computer is called a statement. The `class` statement is the way you give your computer program a name. It's also used to determine other things about the program, as you will see later. The significance of the term `class` is that Java programs also are called classes.

In this example, the program name `Saluton` matches the document's filename, `Saluton.java`. A Java program must have a name that matches the first part of its filename and should be capitalized the same way.

If the program name doesn't match the filename, you get an error when you try to compile some Java programs, depending on how the `class` statement is being used to configure the program.

What the `main` Statement Does

The next line of the program is the following:

```
public static void main(String[] arguments) {
```

This line tells the computer, "The main part of the program begins here." Java programs are organized into different sections, so there needs to be a way to identify the part of a program that is executed first when the program is run.

The `main` statement is the entry point to most Java programs. The exceptions are applets, programs that are run on a web page by a web browser; servlets, programs run by a web server; and apps, programs run by a mobile device.

Most programs you write during upcoming hours use `main` as their starting point. That's because you run them directly on your computer. Applets, apps, and servlets are run indirectly by another program or device.

To differentiate them from these other types, the programs that you run directly are called applications.

Those Squiggly Bracket Marks

In the `Saluton` program, Lines 3, 4, 6, and 7 contain a squiggly bracket mark of some kind—either a { or a }. These brackets are a way to group lines of your program (in the same way that parentheses are used in a sentence to group words). Everything between the opening bracket { and the closing bracket } is part of the same group.

These groupings are called blocks. In Listing 2.1, the opening bracket on Line 3 is associated with the closing bracket on Line 7, which makes your entire program a block. You use brackets in this way to show the beginning and end of a program.

Blocks can be located inside other blocks (just as parentheses are used in this sentence (and a second set is used here)). The `Saluton` program has brackets on Line 4 and Line 6 that establish another block.

TIP

NetBeans can help you figure out where a block begins and ends. Click one of the brackets in the source code of the `Saluton` program. The bracket you clicked turns yellow along with its corresponding bracket. The Java statements enclosed within the two yellow brackets are a block. This tip is not that useful on a short program like `Saluton`, but as you write much longer programs, it helps you avoid looking like a blockhead.

This block begins with the `main` statement. The lines inside the `main` statement's block will be run when the program begins.

The following statement is the only thing located inside the block:

```
// My first Java program goes here
```

This line is a placeholder. The `//` at the beginning of the line tells the computer to ignore this line because it was put in the program solely for the benefit of humans who are looking at the source code. Lines that serve this purpose are called comments.

Right now, you have written a complete Java program. It can be compiled, but if you run it, nothing happens. The reason is that you haven't told the computer to do anything yet. The `main` statement block contains only a single comment, which is ignored by the computer. You must add some statements inside the opening and closing brackets of the `main` block.

Storing Information in a Variable

In the programs you write, you need a place to store information for a brief period of time. You can do this by using a variable, a storage place that can hold information such as integers, floating-point numbers, true-false values, characters, and lines of text. The information stored in a variable can change, which is how it gets the name variable.

In the `Saluton.java` file, replace Line 5 with the following:

```
String greeting = "Saluton mondo!";
```

This statement tells the computer to store the text "Saluton mondo!" in a variable called `greeting`.

In a Java program, you must tell the computer what type of information a variable will hold. In this program, `greeting` is a string—a line of text that can include letters, numbers, punctuation, and other characters. Putting `String` in the statement sets up the variable to hold string values.

When you enter this statement into the program, a semicolon must be included at the end of the line. Semicolons end each statement in a Java program. They're like the period at the end of a sentence. The computer uses them to determine when one statement ends and the next one begins.

Putting only one statement on each line makes a program more understandable (for us humans).

Displaying the Contents of a Variable

If you run the program at this point, it still seems like nothing happens. The command to store text in the `greeting` variable occurs behind the scenes. To make the computer show that it is doing something, you can display the contents of that variable.

Insert another blank line in the `Saluton` program after the `String greeting = "Saluton mondo!"` statement. Use that empty space to enter the following statement:

```
System.out.println(greeting);
```

This statement tells the computer to display the value stored in the `greeting` variable. The `System.out.println` statement makes the computer display information on the system output device—your monitor.

Now you're getting somewhere.

Saving the Finished Product

Your program should now resemble Listing 2.2, although you might have used slightly different spacing in Lines 5–6. Make any corrections that are needed and save the file (by choosing File, *Save*).

LISTING 2.2 The Finished Version of the `Saluton` Program

```
1: package com.java24hours;
2:
3: class Saluton {
4:     public static void main(String[] arguments) {
5:         String greeting = "Saluton mondo!";
6:         System.out.println(greeting);
7:     }
8: }
```

When the computer runs this program, it runs each of the statements in the `main` statement block on Lines 5 and 6. Listing 2.3 shows what the program would look like if it was written in the English language instead of Java.

LISTING 2.3 A Line-by-Line Breakdown of the `Saluton` Program

```
1: Put this program in the com.java24hours package.
2:
3: The Saluton program begins here:
4:     The main part of the program begins here:
5:         Store the text "Saluton mondo!" in a String variable named
           ➥ greeting
6:         Display the contents of the variable greeting
7:     The main part of the program ends here.
8: The Saluton program ends here.
```

Listing 2.4 shows what the program would look like if written in Klingon, the language of the warrior race from *Star Trek*.

LISTING 2.4 The `Saluton` Program in Klingon

```
1: This program belongs to the house of com.java2hours!
2:
3: Begin the Saluton program here if you know what's good for you!
4:     The main part of the program begins here with honor!
5:         Store the gibberish "Saluton mondo!" in a String variable
           ➥ called greeting!
6:         Display this gibberish from a tongue inferior to Klingon!
7:     End the main part of the program here to avoid my wrath!
8: End the Saluton program now and be grateful you were spared!
```

Compiling the Program into a Class File

Before you can run a Java program, you must compile it. When you compile a program, the instructions given to the computer in the program are converted into a form the computer can better understand.

NetBeans compiles programs automatically as they are saved. If you typed everything as shown in Listing 2.2, the program compiles successfully.

A compiled version of the program, a new file called `Saluton.class`, is created. All Java programs are compiled into class files, which are given the `.class` file extension. A Java program can be made up of several classes that work together, but in a simple program such as `Saluton` only one class is needed.

The compiler turns Java source code into bytecode, a form that can be run by the Java Virtual Machine (JVM).

Fixing Errors

As you compose a program in the NetBeans source editor, errors are flagged with a red alert icon to the left of the editor pane, as shown in Figure 2.2.

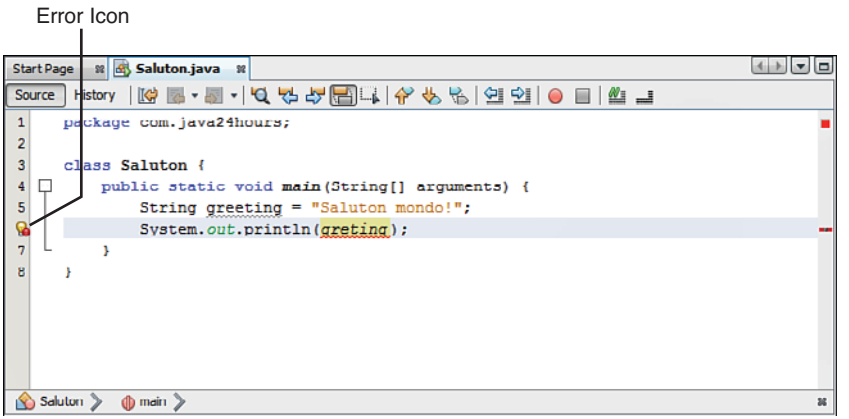

FIGURE 2.2
Spotting errors in the source editor.

The icon appears on the line that triggered the error. You can click this icon to display an error message that explains the compiler error with these details:

- The name of the Java program
- The type of error
- The line where the error was found

Here's an example of an error message you might see when compiling the `Saluton` program:

```
cannot find symbol.
symbol  : variable greting
location: class Saluton
```

The error is the first line of the message: "cannot find symbol." These messages often can be confusing to new programmers. When the error message doesn't make sense to you, don't spend much time trying to figure it out. Instead, take a look at the line where the error occurred and look for the most obvious causes.

NOTE

The Java compiler speaks up only when there's an error to complain about. If you compile a program successfully without any errors, nothing happens in response. This is anticlimactic. When I was starting out as a Java programmer, I was hoping successful compilation would be met with a grand flourish of celebratory horns.

TIP

This book's official website at www.java24hours.com includes source files for all programs you create. If you can't find any typos or other reasons for errors in the `Saluton` program but there are still errors, go to the book's website and download `Saluton.java` from the Hour 2 page. Try to run that file instead.

For instance, can you determine what's wrong with the following statement?

```
System.out.println(greting);
```

The error is a typo in the variable name, which should be `greeting` instead of `greting`. (Add this typo on purpose in NetBeans to see what happens.)

If you get error messages when creating the `Saluton` program, double-check that your program matches Listing 2.2 and correct any differences you find. Make sure that everything is capitalized correctly and all punctuation marks such as {, }, and ; are included.

Often, a close look at the line identified by the error message is enough to reveal the error (or errors) that need to be fixed.

Running a Java Program

To see whether the `Saluton` program does what you want, run the class with the Java Virtual Machine, the interpreter that runs all Java code. In NetBeans, choose the menu command Run, Run File. An Output pane opens below the source code editor. In this pane, if there are no errors, the program displays the output, as shown in Figure 2.3.

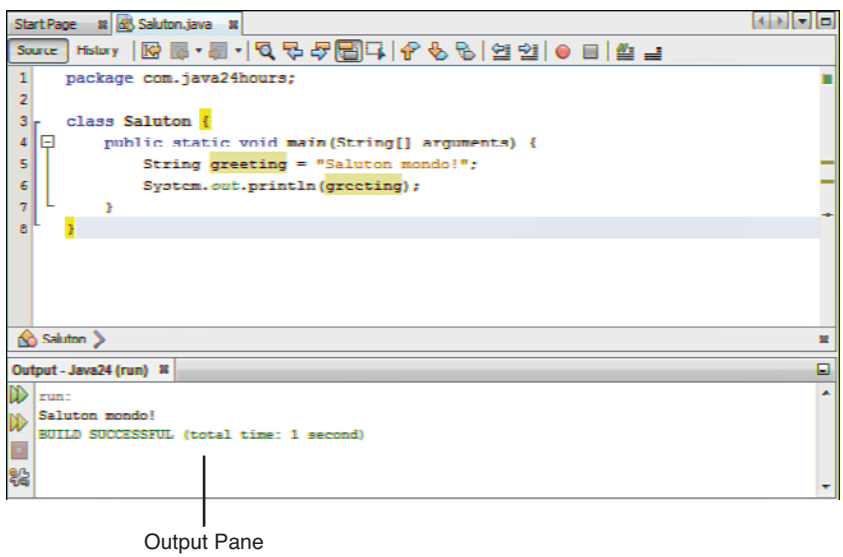

FIGURE 2.3
Running your first Java program.

If you see the text "Saluton Mondo!" you have just written your first working Java program! Your computer has just greeted the world—a tradition in the computer programming field that's as important to many of us as caffeine, short-sleeved dress shirts, and *Call of Duty*.

You might be asking yourself why "Saluton mondo!" is a traditional greeting. The phrase means "Hello world!" in Esperanto, an artificial language created by Ludwig Zamenhof in 1887 to facilitate international communication. It's only a traditional greeting in the sense that I'm trying to start that tradition.

NOTE

Oracle offers comprehensive documentation for the Java language on the Web. You don't need it to use this book, because each topic is discussed fully as it is introduced, but this reference comes in handy when you want to expand your knowledge and write your own programs.

The documentation can be downloaded, but it's more convenient to browse as needed on Oracle's website. The most up-to-date Java documentation is available at http://download.java.net/jdk8/docs/api.

Summary

During this hour, you got your first chance to create a Java program. You learned that to develop a Java program you need to complete these four basic steps:

1. Write the program with a text editor or a tool such as NetBeans.
2. Compile the program into a class file.
3. Tell the Java Virtual Machine to run the class.
4. Call your mother.

Along the way, you were introduced to some basic computer programming concepts such as compilers, interpreters, blocks, statements, and variables. These will become clearer to you in successive hours. As long as you got the `Saluton` program to work during this hour, you're ready to proceed.

(The fourth step has nothing to do with Java programming. It's just something my mother suggested I put in the book.)

Workshop

Q&A

Q. How important is it to put the right number of blank spaces on a line in a Java program?

A. It's completely unimportant as far as the computer is concerned. Spacing is strictly for the benefit of people looking at a computer program—the Java compiler couldn't care less. You could have written the `Saluton` program without using blank spaces or used the Tab key to indent lines, and it would compile successfully.

Although the number of spaces in front of lines isn't important, you should use consistent spacing and indentation in your Java programs. Why? Because spacing makes it easier for you to see how a program is organized and to which programming block a statement belongs.

The programs you write must be understandable to other programmers, including yourself when you look at the code weeks or months later to fix a bug or make an enhancement. Consistency in spacing and indentation are part of what's called a programming style. Good programmers adopt a style and practice it in all their work.

Q. A Java program has been described as a class and as a group of classes. Which is it?

A. Both. The simple Java programs you create during the next few hours are compiled into a single file with the extension `.class`. You can run these with the Java Virtual Machine. Java programs also can be made up of a set of classes that work together. This topic is fully explored during Hour 10, "Creating Your First Object."

Q. If semicolons are needed at the end of each statement, why does the comment line `// My first Java program goes here` not end with a semicolon?

A. Comments are completely ignored by the compiler. If you put `//` on a line in your program, this tells the Java compiler to ignore everything to the right of the `//` on that line. The following example shows a comment on the same line as a statement:

```
System.out.println(greeting); // hello, world!
```

Q. I couldn't find any errors in the line where the compiler noted an error. What can I do?

A. The line number displayed with the error message isn't always the place where an error needs to be fixed. Examine the statements that are directly above the error message to see whether you can spot any typos or other bugs. The error usually is within the same programming block.

Q. How can I visit Antarctica?

A. If you're not willing to become a scientific researcher or a support staffer such as a cook, electrician, or doctor, you can become one of the 10,000 people who visit the frozen continent annually as tourists.

Flyovers are available from Australia, New Zealand, and South America and cost around $1,000 per person.

Several cruise ships visit for a trip lasting from 10 days to three weeks, the most expensive of which is around $25,000. Some cruises offer a chance to kayak or hike among penguins, visit icebergs, and even camp overnight.

The Polar Cruises website at www.polarcruises.com provides more information for prospective Antarctica visitors.

The British Antarctic Survey offers a piece of advice for visitors: "Do not walk onto glaciers or large snowfields unless properly trained."

Quiz

Test your knowledge of the material covered in this hour by answering the following questions.

1. When you compile a Java program, what are you doing?

A. Saving it to a disk

B. Converting it into a form the computer can better understand

C. Adding it to your program collection

2. What is a variable?

A. Something that wobbles but doesn't fall down

B. Text in a program that the compiler ignores

C. A place to store information in a program

3. What is the process of fixing errors called?
   - **A.** Defrosting
   - **B.** Debugging
   - **C.** Decomposing

Answers

1. **B.** Compiling a program converts a `.java` file into a `.class` file or a set of `.class` files.
2. **C.** Variables are one place to store information; later you learn about others such as arrays and constants. Weebles wobble but they don't fall down, and comments are text in a program that the compiler ignores.
3. **B.** Because errors in a computer program are called bugs, fixing those errors is called debugging. Some programming tools come with a tool called a debugger that helps you fix errors. NetBeans has one of debest debuggers.

Activities

If you'd like to explore the topics covered in this hour a little more fully, try the following activities:

- You can translate the English phrase "Hello world!" into other languages using the Google Translator at http://translate.google.com. Write a program that enables your computer to greet the world in a language such as French, Italian, or Portuguese.
- Go back to the `Saluton` program and add one or two errors. For example, take a semicolon off the end of a line or change the text `println` on one line to `print1n` (with a number 1 instead of the letter L). Save the program and try to compile it; then compare the error messages you see to the errors you caused.

To see solutions to these activities, visit the book's website at www.java24hours.com.

INDEX

NUMERICS

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X