

starting out with >>>

JAVA™

From Control Structures through Objects

6TH EDITION



TONY GADDIS

STARTING OUT WITH

JAVA™



From Control Structures
through Objects

This page intentionally left blank

STARTING OUT WITH

JAVA™

From Control Structures
through Objects

SIXTH EDITION

Tony Gaddis

Haywood Community College

PEARSON

Boston Columbus Indianapolis New York San Francisco Hoboken
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montréal Toronto
Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Editor in Chief: Marcia Horton
Acquisitions Editor: Matt Goldstein
Editorial Assistant: Kelsey Loanes
VP of Marketing: Christy Lesko
Director of Field Marketing: Tim Galligan
Product Marketing Manager: Bram van Kempen
Field Marketing Manager: Demetrius Hall
Marketing Assistant: Jon Bryant
Director of Product Management: Erin Gregg
Team Lead Product Management: Scott Disanno
Program Manager: Carole Snyder
Production Project Manager: Camille Trentacoste
Procurement Manager: Mary Fischer

Senior Specialist, Program Planning and Support:
Maura Zaldivar-Garcia
Cover Designer: Joyce Wells
Cover Image: Binh Thanh Bui/Shutterstock
Manager, Rights Management: Rachel Youdelman
Associate Project Manager, Rights Management:
William J. Opaluch
Full-Service Project Management: Kailash Jadli,
Aptara®, Inc.
Composition: Aptara®, Inc.
Printer/Bindery: Edwards Brothers
Cover printer: Phoenix Color/Hagerstown

Copyright © 2016 by Pearson Education, Inc. or its affiliates. All Rights Reserved. Printed in the United States of America. This publication is protected by copyright, and permission should be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise. For information regarding permissions, request forms and the appropriate contacts within the Pearson Education Global Rights & Permissions department, please visit www.pearsoned.com/permissions/.

Many of the designations by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed in initial caps or all caps.

Unless otherwise indicated herein, any third-party trademarks that may appear in this work are the property of their respective owners and any references to third-party trademarks, logos or other trade dress are for demonstrative or descriptive purposes only. Such references are not intended to imply any sponsorship, endorsement, authorization, or promotion of Pearson's products by the owners of such marks, or any relationship between the owner and Pearson Education, Inc. or its affiliates, authors, licensees or distributors.

Microsoft and/or its respective suppliers make no representations about the suitability of the information contained in the documents and related graphics published as part of the services for any purpose. All such documents and related graphics are provided "as is" without warranty of any kind. Microsoft and/or its respective suppliers hereby disclaim all warranties and conditions with regard to this information, including all warranties and conditions of merchantability. Whether express, implied or statutory, fitness for a particular purpose, title and non-infringement. In no event shall Microsoft and/or its respective suppliers be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract. Negligence or other tortious action, arising out of or in connection with the use or performance of information available from the services.

The documents and related graphics contained herein could include technical inaccuracies or typographical errors changes are periodically added to the information herein. Microsoft and/or its respective suppliers may make improvements and/or changes in the product(s) and/or the program(s) described herein at any time partial screen shots may be viewed in full within the software version specified.

Microsoft® Windows®, and Microsoft Office® are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation.

Library of Congress Cataloging-in-Publication Data

Gaddis, Tony, author.

Starting out with Java. From control structures through objects/Tony Gaddis, Haywood Community College.—6th edition.

pages cm

ISBN-13: 978-0-13-395705-1

ISBN-10: 0-13-395705-5

1. Java (Computer program language)
2. Data structures (Computer science)
3. Object-oriented programming (Computer science)

QA76.73.J38G333 2016

005.13'3—dc23

2014049102

10 9 8 7 6 5 4 3 2 1

PEARSON

ISBN-13: 978-0-13-395705-1
ISBN-10: 0-13-395705-5

Contents in Brief

	Preface	xxiii	
Chapter 1	Introduction to Computers and Java		1
Chapter 2	Java Fundamentals		27
Chapter 3	Decision Structures		111
Chapter 4	Loops and Files		189
Chapter 5	Methods		269
Chapter 6	A First Look at Classes		319
Chapter 7	Arrays and the <code>ArrayList</code> Class		405
Chapter 8	A Second Look at Classes and Objects		495
Chapter 9	Text Processing and More about Wrapper Classes		559
Chapter 10	Inheritance		613
Chapter 11	Exceptions and Advanced File I/O		703
Chapter 12	A First Look at GUI Applications		761
Chapter 13	Advanced GUI Applications		849
Chapter 14	Applets and More		917
Chapter 15	Creating GUI Applications with JavaFX and Scene Builder		991
Chapter 16	Recursion		1047
Chapter 17	Databases		1075
	Index		1171
	Appendixes A–M		Companion Website
	Case Studies 1–7		Companion Website

This page intentionally left blank

Contents

Preface xxiii

Chapter 1 **Introduction to Computers and Java** **1**

1.1	Introduction	1
1.2	Why Program?	1
1.3	Computer Systems: Hardware and Software	2
	<i>Hardware</i>	2
	<i>Software</i>	5
1.4	Programming Languages	6
	<i>What Is a Program?</i>	6
	<i>A History of Java</i>	8
	<i>Java Applications and Applets</i>	8
1.5	What Is a Program Made Of?	9
	<i>Language Elements</i>	9
	<i>Lines and Statements</i>	11
	<i>Variables</i>	11
	<i>The Compiler and the Java Virtual Machine</i>	12
	<i>Java Software Editions</i>	14
	<i>Compiling and Running a Java Program</i>	14
1.6	The Programming Process	16
	<i>Software Engineering</i>	18
1.7	Object-Oriented Programming	19
	<i>Review Questions and Exercises</i> 21	
	<i>Programming Challenge</i> 25	

Chapter 2 **Java Fundamentals** **27**

2.1	The Parts of a Java Program	27
2.2	The <code>print</code> and <code>println</code> Methods, and the Java API	33
2.3	Variables and Literals	39
	<i>Displaying Multiple Items with the + Operator</i>	40
	<i>Be Careful with Quotation Marks</i>	41
	<i>More about Literals</i>	42

	<i>Identifiers</i>	42
	<i>Class Names</i>	44
2.4	Primitive Data Types	44
	<i>The Integer Data Types</i>	46
	<i>Floating-Point Data Types</i>	47
	<i>The boolean Data Type</i>	50
	<i>The char Data Type</i>	50
	<i>Variable Assignment and Initialization</i>	52
	<i>Variables Hold Only One Value at a Time</i>	53
2.5	Arithmetic Operators	54
	<i>Integer Division</i>	57
	<i>Operator Precedence</i>	57
	<i>Grouping with Parentheses</i>	59
	<i>The Math Class</i>	62
2.6	Combined Assignment Operators	63
2.7	Conversion between Primitive Data Types	65
	<i>Mixed Integer Operations</i>	67
	<i>Other Mixed Mathematical Expressions</i>	68
2.8	Creating Named Constants with <code>final</code>	69
2.9	The <code>String</code> Class	70
	<i>Objects Are Created from Classes</i>	70
	<i>The <code>String</code> Class</i>	71
	<i>Primitive Type Variables and Class Type Variables</i>	71
	<i>Creating a <code>String</code> Object</i>	72
2.10	Scope	75
2.11	Comments	77
2.12	Programming Style	82
2.13	Reading Keyboard Input	84
	<i>Reading a Character</i>	88
	<i>Mixing Calls to <code>nextLine</code> with Calls to Other <code>Scanner</code> Methods</i>	88
2.14	Dialog Boxes	92
	<i>Displaying Message Dialogs</i>	92
	<i>Displaying Input Dialogs</i>	93
	<i>An Example Program</i>	93
	<i>Converting String Input to Numbers</i>	95
2.15	Common Errors to Avoid	99
	<i>Review Questions and Exercises</i> 100	
	<i>Programming Challenges</i> 105	

Chapter 3 **Decision Structures** 111

3.1	The <code>if</code> Statement	111
	<i>Using Relational Operators to Form Conditions</i>	113
	<i>Putting It All Together</i>	114
	<i>Programming Style and the <code>if</code> Statement</i>	118
	<i>Be Careful with Semicolons</i>	119

	<i>Having Multiple Conditionally Executed Statements</i>	119
	<i>Flags</i>	120
	<i>Comparing Characters</i>	120
3.2	The <code>if-else</code> Statement	121
3.3	Nested <code>if</code> Statements	124
3.4	The <code>if-else-if</code> Statement	131
3.5	Logical Operators	137
	<i>The Precedence of Logical Operators</i>	143
	<i>Checking Numeric Ranges with Logical Operators</i>	144
3.6	Comparing <code>String</code> Objects	145
	<i>Ignoring Case in String Comparisons</i>	150
3.7	More about Variable Declaration and Scope	151
3.8	The Conditional Operator (Optional)	152
3.9	The <code>switch</code> Statement	154
3.10	Displaying Formatted Output with <code>System.out.printf</code> and <code>String.format</code>	164
	<i>Format Specifier Syntax</i>	167
	<i>Precision</i>	167
	<i>Specifying a Minimum Field Width</i>	168
	<i>Flags</i>	170
	<i>Formatting String Arguments</i>	174
	<i>The <code>String.format</code> Method</i>	175
3.11	Common Errors to Avoid	178
	<i>Review Questions and Exercises</i>	179
	<i>Programming Challenges</i>	184

Chapter 4 **Loops and Files** 189

4.1	The Increment and Decrement Operators	189
	<i>The Difference between Postfix and Prefix Modes</i>	192
4.2	The <code>while</code> Loop	193
	<i>The <code>while</code> Loop Is a Pretest Loop</i>	196
	<i>Infinite Loops</i>	196
	<i>Don't Forget the Braces with a Block of Statements</i>	197
	<i>Programming Style and the <code>while</code> Loop</i>	198
4.3	Using the <code>while</code> Loop for Input Validation	200
4.4	The <code>do-while</code> Loop	204
4.5	The <code>for</code> Loop	207
	<i>The <code>for</code> Loop Is a Pretest Loop</i>	210
	<i>Avoid Modifying the Control Variable in the Body of the <code>for</code> Loop</i>	211
	<i>Other Forms of the Update Expression</i>	211
	<i>Declaring a Variable in the <code>for</code> Loop's Initialization Expression</i>	211
	<i>Creating a User Controlled <code>for</code> Loop</i>	212
	<i>Using Multiple Statements in the Initialization and Update Expressions</i>	213

4.6 Running Totals and Sentinel Values. 216
Using a Sentinel Value 219

4.7 Nested Loops. 221

4.8 The `break` and `continue` Statements (Optional) 229

4.9 Deciding Which Loop to Use 229

4.10 Introduction to File Input and Output 230
Using the `PrintWriter` Class to Write Data to a File. 230
Appending Data to a File 236
Specifying the File Location 237
Reading Data from a File 237
Reading Lines from a File with the `nextLine` Method 238
Adding a `throws` Clause to the Method Header 241
Checking for a File's Existence 245

4.11 Generating Random Numbers with the `Random` Class. 249

4.12 Common Errors to Avoid 255

Review Questions and Exercises 256

Programming Challenges 262

Chapter 5 Methods 269

5.1 Introduction to Methods 269
void Methods and Value-Returning Methods 270
Defining a void Method 271
Calling a Method. 272
Hierarchical Method Calls 277
Using Documentation Comments with Methods. 278

5.2 Passing Arguments to a Method. 279
Argument and Parameter Data Type Compatibility. 281
Parameter Variable Scope 282
Passing Multiple Arguments. 282
Arguments Are Passed by Value 284
Passing Object References to a Method 285
Using the `@param` Tag in Documentation Comments. 288

5.3 More about Local Variables 291
Local Variable Lifetime 292
Initializing Local Variables with Parameter Values. 292

5.4 Returning a Value from a Method. 293
Defining a Value-Returning Method. 293
Calling a Value-Returning Method 295
Using the `@return` Tag in Documentation Comments. 296
Returning a boolean Value. 300
Returning a Reference to an Object 300

5.5 Problem Solving with Methods 302
Calling Methods That Throw Exceptions. 306

5.6 Common Errors to Avoid 306

Review Questions and Exercises 307

Programming Challenges 312

Chapter 6	A First Look at Classes	319
6.1	Objects and Classes	319
	<i>Classes: Where Objects Come From</i>	320
	<i>Classes in the Java API</i>	321
	<i>Primitive Variables vs. Objects</i>	323
6.2	Writing a Simple Class, Step by Step	326
	<i>Accessor and Mutator Methods</i>	340
	<i>The Importance of Data Hiding</i>	340
	<i>Avoiding Stale Data</i>	341
	<i>Showing Access Specification in UML Diagrams</i>	341
	<i>Data Type and Parameter Notation in UML Diagrams</i>	341
	<i>Layout of Class Members</i>	342
6.3	Instance Fields and Methods	343
6.4	Constructors	348
	<i>Showing Constructors in a UML Diagram</i>	350
	<i>Uninitialized Local Reference Variables</i>	350
	<i>The Default Constructor</i>	350
	<i>Writing Your Own No-Arg Constructor</i>	351
	<i>The String Class Constructor</i>	352
6.5	Passing Objects as Arguments	360
6.6	Overloading Methods and Constructors	372
	<i>The BankAccount Class</i>	374
	<i>Overloaded Methods Make Classes More Useful</i>	380
6.7	Scope of Instance Fields	380
	<i>Shadowing</i>	381
6.8	Packages and <i>import</i> Statements	382
	<i>Explicit and Wildcard import Statements</i>	382
	<i>The java.lang Package</i>	383
	<i>Other API Packages</i>	383
6.9	Focus on Object-Oriented Design: Finding the Classes and Their Responsibilities	384
	<i>Finding the Classes</i>	384
	<i>Identifying a Class's Responsibilities</i>	387
	<i>This Is Only the Beginning</i>	390
6.10	Common Errors to Avoid	390
	<i>Review Questions and Exercises</i>	391
	<i>Programming Challenges</i>	396
Chapter 7	Arrays and the ArrayList Class	405
7.1	Introduction to Arrays	405
	<i>Accessing Array Elements</i>	407
	<i>Inputting and Outputting Array Contents</i>	408
	<i>Java Performs Bounds Checking</i>	411
	<i>Watch Out for Off-by-One Errors</i>	412
	<i>Array Initialization</i>	413
	<i>Alternate Array Declaration Notation</i>	414

7.2	Processing Array Elements	415
	<i>Array Length</i>	417
	<i>The Enhanced for Loop</i>	418
	<i>Letting the User Specify an Array's Size</i>	419
	<i>Reassigning Array Reference Variables</i>	421
	<i>Copying Arrays</i>	422
7.3	Passing Arrays as Arguments to Methods	424
7.4	Some Useful Array Algorithms and Operations	428
	<i>Comparing Arrays</i>	428
	<i>Summing the Values in a Numeric Array</i>	429
	<i>Getting the Average of the Values in a Numeric Array</i>	430
	<i>Finding the Highest and Lowest Values in a Numeric Array</i>	430
	<i>The SalesData Class</i>	431
	<i>Partially Filled Arrays</i>	439
	<i>Working with Arrays and Files</i>	440
7.5	Returning Arrays from Methods	441
7.6	string Arrays	443
	<i>Calling string Methods from an Array Element</i>	445
7.7	Arrays of Objects	446
7.8	The Sequential Search Algorithm	449
7.9	Two-Dimensional Arrays	452
	<i>Initializing a Two-Dimensional Array</i>	456
	<i>The length Field in a Two-Dimensional Array</i>	457
	<i>Displaying All the Elements of a Two-Dimensional Array</i>	459
	<i>Summing All the Elements of a Two-Dimensional Array</i>	459
	<i>Summing the Rows of a Two-Dimensional Array</i>	460
	<i>Summing the Columns of a Two-Dimensional Array</i>	460
	<i>Passing Two-Dimensional Arrays to Methods</i>	461
	<i>Ragged Arrays</i>	463
7.10	Arrays with Three or More Dimensions	464
7.11	The Selection Sort and the Binary Search Algorithms	465
	<i>The Selection Sort Algorithm</i>	465
	<i>The Binary Search Algorithm</i>	468
7.12	Command-Line Arguments and Variable-Length Argument Lists	470
	<i>Command-Line Arguments</i>	471
	<i>Variable-Length Argument Lists</i>	472
7.13	The ArrayList Class	474
	<i>Creating and Using an ArrayList Object</i>	475
	<i>Using the Enhanced for Loop with an ArrayList</i>	476
	<i>The ArrayList Class's toString method</i>	477
	<i>Removing an Item from an ArrayList</i>	478
	<i>Inserting an Item</i>	479
	<i>Replacing an Item</i>	480
	<i>Capacity</i>	481
	<i>Using the Diamond Operator for Type Inference (Java 7)</i>	482
7.14	Common Errors to Avoid	483

Review Questions and Exercises 483
Programming Challenges 488

Chapter 8 A Second Look at Classes and Objects 495

8.1 Static Class Members 495
A Quick Review of Instance Fields and Instance Methods 495
Static Members 496
Static Fields 496
Static Methods 499

8.2 Passing Objects as Arguments to Methods 502

8.3 Returning Objects from Methods 505

8.4 The `toString` Method 507

8.5 Writing an `equals` Method 511

8.6 Methods That Copy Objects 514
Copy Constructors 516

8.7 Aggregation 517
Aggregation in UML Diagrams 525
Security Issues with Aggregate Classes 525
Avoid Using null References 527

8.8 The `this` Reference Variable 530
Using this to Overcome Shadowing 531
Using this to Call an Overloaded Constructor from Another Constructor 532

8.9 Enumerated Types 533
Enumerated Types Are Specialized Classes 534
Switching On an Enumerated Type 540

8.10 Garbage Collection 542
The finalize Method 544

8.11 Focus on Object-Oriented Design: Class Collaboration 544
Determining Class Collaborations with CRC Cards 547

8.12 Common Errors to Avoid 548
Review Questions and Exercises 549
Programming Challenges 553

Chapter 9 Text Processing and More about Wrapper Classes 559

9.1 Introduction to Wrapper Classes 559

9.2 Character Testing and Conversion with the `Character` Class 560
Character Case Conversion 565

9.3 More `String` Methods 568
Searching for Substrings 568
Extracting Substrings 575
Methods That Return a Modified String 579
The Static `valueOf` Methods 580

9.4 The `StringBuilder` Class 582
The `StringBuilder` Constructors 583
Other `StringBuilder` Methods 584
The `toString` Method 587

9.5 Tokenizing Strings 593

9.6 Wrapper Classes for the Numeric Data Types 597
The Static `toString` Methods 598
The `toBinaryString`, `toHexString`, and `toOctalString` Methods 598
The `MIN_VALUE` and `MAX_VALUE` Constants 598
Autoboxing and Unboxing 598

9.7 Focus on Problem Solving: The `TestScoreReader` Class 600

9.8 Common Errors to Avoid 604
Review Questions and Exercises 605
Programming Challenges 608

Chapter 10 **Inheritance** 613

10.1 What Is Inheritance? 613
Generalization and Specialization 613
Inheritance and the “Is a” Relationship 614
Inheritance in UML Diagrams 622
The Superclass’s Constructor 623
Inheritance Does Not Work in Reverse 625

10.2 Calling the Superclass Constructor 626
*When the Superclass Has No Default
or No-Arg Constructors* 632
Summary of Constructor Issues in Inheritance 633

10.3 Overriding Superclass Methods 634
Overloading versus Overriding 639
Preventing a Method from Being Overridden 642

10.4 Protected Members 643
Package Access 648

10.5 Chains of Inheritance 649
Class Hierarchies 655

10.6 The `Object` Class 655

10.7 Polymorphism 657
Polymorphism and Dynamic Binding 658
The “Is-a” Relationship Does Not Work in Reverse 660
The `instanceof` Operator 661

10.8 Abstract Classes and Abstract Methods 662
Abstract Classes in UML 668

10.9 Interfaces 669
An Interface is a Contract 671
Fields in Interfaces 675
Implementing Multiple Interfaces 675
Interfaces in UML 675

	<i>Default Methods</i>	676
	<i>Polymorphism and Interfaces</i>	678
10.10	Anonymous Inner Classes	683
10.11	Functional Interfaces and Lambda Expressions	686
10.12	Common Errors to Avoid	691
	<i>Review Questions and Exercises</i>	692
	<i>Programming Challenges</i>	698

Chapter 11 **Exceptions and Advanced File I/O** 703

11.1	Handling Exceptions	703
	<i>Exception Classes</i>	704
	<i>Handling an Exception</i>	705
	<i>Retrieving the Default Error Message</i>	709
	<i>Polymorphic References to Exceptions</i>	712
	<i>Using Multiple catch Clauses to Handle Multiple Exceptions</i>	712
	<i>The finally Clause</i>	720
	<i>The Stack Trace</i>	722
	<i>Handling Multiple Exceptions with One catch Clause (Java 7)</i>	723
	<i>When an Exception Is Not Caught</i>	725
	<i>Checked and Unchecked Exceptions</i>	726
11.2	Throwing Exceptions	727
	<i>Creating Your Own Exception Classes</i>	730
	<i>Using the @exception Tag in Documentation Comments</i>	733
11.3	Advanced Topics: Binary Files, Random Access Files, and Object Serialization	734
	<i>Binary Files</i>	734
	<i>Random Access Files</i>	741
	<i>Object Serialization</i>	746
	<i>Serializing Aggregate Objects</i>	750
11.4	Common Errors to Avoid	751
	<i>Review Questions and Exercises</i>	751
	<i>Programming Challenges</i>	757

Chapter 12 **A First Look at GUI Applications** 761

12.1	Introduction	761
	<i>The JFC, AWT, and Swing</i>	762
	<i>Event-Driven Programming</i>	764
	<i>The javax.swing and java.awt Packages</i>	764
12.2	Creating Windows	764
	<i>Using Inheritance to Extend the JFrame Class</i>	767
	<i>Equipping GUI Classes with a main Method</i>	769
	<i>Adding Components to a Window</i>	771
	<i>Handling Events with Action Listeners</i>	777

	<i>Writing an Event Listener for the <code>KiloConverter</code> Class</i>	779
	<i>Background and Foreground Colors</i>	784
	<i>The <code>ActionEvent</code> Object</i>	788
12.3	Layout Managers	793
	<i>Adding a Layout Manager to a Container</i>	794
	<i>The <code>FlowLayout</code> Manager</i>	794
	<i>The <code>BorderLayout</code> Manager</i>	797
	<i>The <code>GridLayout</code> Manager</i>	804
12.4	Radio Buttons and Check Boxes	810
	<i>Radio Buttons</i>	810
	<i>Check Boxes</i>	816
12.5	Borders	821
12.6	Focus on Problem Solving: Extending Classes from <code>JPanel</code>	824
	<i>The Brandi's Bagel House Application</i>	824
	<i>The <code>GreetingPanel</code> Class</i>	825
	<i>The <code>BagelPanel</code> Class</i>	826
	<i>The <code>ToppingPanel</code> Class</i>	828
	<i>The <code>CoffeePanel</code> Class</i>	830
	<i>Putting It All Together</i>	832
12.7	Splash Screens	836
12.8	Using Console Output to Debug a GUI Application	837
12.9	Common Errors to Avoid	842
	<i>Review Questions and Exercises</i>	842
	<i>Programming Challenges</i>	845

Chapter 13 **Advanced GUI Applications** 849

13.1	The Swing and AWT Class Hierarchy	849
13.2	Read-Only Text Fields	850
13.3	Lists	852
	<i>Selection Modes</i>	852
	<i>Responding to List Events</i>	853
	<i>Retrieving the Selected Item</i>	854
	<i>Placing a Border around a List</i>	858
	<i>Adding a Scroll Bar to a List</i>	858
	<i>Adding Items to an Existing <code>JList</code> Component</i>	863
	<i>Multiple Selection Lists</i>	863
13.4	Combo Boxes	868
	<i>Retrieving the Selected Item</i>	869
13.5	Displaying Images in Labels and Buttons	874
13.6	Mnemonics and Tool Tips	880
	<i>Mnemonics</i>	880
	<i>Tool Tips</i>	882
13.7	File Choosers and Color Choosers	882
	<i>File Choosers</i>	883
	<i>Color Choosers</i>	885

13.8	Menus	886
13.9	More about Text Components: Text Areas and Fonts	895
	<i>Text Areas</i>	895
	<i>Fonts</i>	898
13.10	Sliders	899
13.11	Look and Feel	904
13.12	Common Errors to Avoid	906
	<i>Review Questions and Exercises</i>	907
	<i>Programming Challenges</i>	912

Chapter 14 **Applets and More** 917

14.1	Introduction to Applets	917
14.2	A Brief Introduction to HTML	919
	<i>Hypertext</i>	919
	<i>Markup Language</i>	920
	<i>Document Structure Tags</i>	920
	<i>Text Formatting Tags</i>	922
	<i>Creating Breaks in Text</i>	924
	<i>Inserting Links</i>	927
14.3	Creating Applets with Swing	928
	<i>Running an Applet</i>	930
	<i>Handling Events in an Applet</i>	932
14.4	Using AWT for Portability	937
14.5	Drawing Shapes	942
	<i>The XY Coordinate System</i>	942
	<i>Graphics Objects</i>	942
	<i>The repaint Method</i>	956
	<i>Drawing on Panels</i>	957
14.6	Handling Mouse Events	963
	<i>Handling Mouse Events</i>	963
14.7	Timer Objects	973
14.8	Playing Audio	977
	<i>Using an AudioClip Object</i>	978
	<i>Playing Audio in an Application</i>	981
14.9	Common Errors to Avoid	982
	<i>Review Questions and Exercises</i>	982
	<i>Programming Challenges</i>	988

Chapter 15 **Creating GUI Applications with JavaFX and Scene Builder** 991

15.1	Introduction	991
	<i>Event-Driven Programming</i>	993
15.2	Scene Graphs	993

- 15.3 Using Scene Builder to Create JavaFX Applications 995
 - Starting Scene Builder* 996
 - The Scene Builder Main Window* 997
- 15.4 Writing the Application Code 1009
 - The Main Application Class* 1010
 - The Controller Class*. 1012
 - Using the Sample Controller Skeleton*. 1017
 - Summary of Creating a JavaFX Application* 1018
- 15.5 RadioButtons and CheckBoxes 1019
 - RadioButtons*. 1019
 - Determining in Code Whether a RadioButton Is Selected*. 1021
 - Responding to RadioButton Events*. 1024
 - CheckBoxes* 1027
 - Determining in Code Whether a CheckBox Is Selected*. 1028
 - Responding to CheckBox Events* 1030
- 15.6 Displaying Images 1033
 - Displaying an Image with Code* 1034
- 15.7 Common Errors to Avoid 1038
 - Review Questions and Exercises* 1038
 - Programming Challenges* 1042

Chapter 16 Recursion 1047

- 16.1 Introduction to Recursion. 1047
- 16.2 Solving Problems with Recursion 1050
 - Direct and Indirect Recursion* 1054
- 16.3 Examples of Recursive Methods 1055
 - Summing a Range of Array Elements with Recursion*. 1055
 - Drawing Concentric Circles* 1056
 - The Fibonacci Series*. 1058
 - Finding the Greatest Common Divisor* 1060
- 16.4 A Recursive Binary Search Method 1061
- 16.5 The Towers of Hanoi 1064
- 16.6 Common Errors to Avoid 1069
 - Review Questions and Exercises* 1069
 - Programming Challenges* 1072

Chapter 17 Databases 1075

- 17.1 Introduction to Database Management Systems 1075
 - JDBC* 1076
 - SQL*. 1077
 - Using a DBMS*. 1077
 - Java DB*. 1078
 - Creating the *coffeeDB* Database* 1078

	<i>Connecting to the coffeeDB Database</i>	1078
	<i>Connecting to a Password-Protected Database</i>	1080
17.2	Tables, Rows, and Columns	1081
	<i>Column Data Types</i>	1083
	<i>Primary Keys</i>	1083
17.3	Introduction to the SQL <code>SELECT</code> Statement	1084
	<i>Passing an SQL Statement to the DBMS</i>	1086
	<i>Specifying Search Criteria with the <code>WHERE</code> Clause</i>	1096
	<i>Sorting the Results of a <code>SELECT</code> Query</i>	1102
	<i>Mathematical Functions</i>	1103
17.4	Inserting Rows	1106
	<i>Inserting Rows with JDBC</i>	1108
17.5	Updating and Deleting Existing Rows	1110
	<i>Updating Rows with JDBC</i>	1111
	<i>Deleting Rows with the <code>DELETE</code> Statement</i>	1115
	<i>Deleting Rows with JDBC</i>	1115
17.6	Creating and Deleting Tables	1119
	<i>Removing a Table with the <code>DROP TABLE</code> Statement</i>	1122
17.7	Creating a New Database with JDBC	1122
17.8	Scrollable Result Sets	1124
17.9	Result Set Metadata	1125
17.10	Displaying Query Results in a <code>JTable</code>	1129
17.11	Relational Data	1139
	<i>Joining Data from Multiple Tables</i>	1142
	<i>An Order Entry System</i>	1143
17.12	Advanced Topics	1161
	<i>Transactions</i>	1161
	<i>Stored Procedures</i>	1162
17.13	Common Errors to Avoid	1163
	<i>Review Questions and Exercises</i>	1163
	<i>Programming Challenges</i>	1168

Index 1171

Companion Website:

Appendix A	Working with Records and Random Access Files
Appendix B	The ASCII/Unicode Characters
Appendix C	Operator Precedence and Associativity
Appendix D	Java Key Words
Appendix E	Installing the JDK and JDK Documentation
Appendix F	Using the <code>javadoc</code> Utility
Appendix G	More about the <code>Math</code> Class
Appendix H	Packages
Appendix I	More about <code>JOptionPane</code> Dialog Boxes
Appendix J	Answers to Checkpoints
Appendix K	Answers to Odd-Numbered Review Questions

Appendix L	Getting Started with Alice
Appendix M	Configuring JavaDB
Case Study 1	Calculating Sales Commission
Case Study 2	The Amortization Class
Case Study 3	The PinTester Class
Case Study 4	Parallel Arrays
Case Study 5	The FeetInches Class
Case Study 6	The SerialNumber Class
Case Study 7	A Simple Text Editor Application

LOCATION OF VIDEONOTES IN THE TEXT



Chapter 1	Compiling and Running a Java Program, p. 14 Using an IDE, p. 15 Your First Java Program, p. 25
Chapter 2	Displaying Console Output, p. 33 Declaring Variables, p. 39 Simple Math Expressions, p. 55 The Miles-per-Gallon Problem, p. 106
Chapter 3	The <code>if</code> Statement, p. 111 The <code>if-else</code> Statement, p. 121 The <code>if-else-if</code> Statement, p. 132 The Time Calculator Problem, p. 185
Chapter 4	The <code>while</code> Loop, p. 193 The Pennies for Pay Problem, p. 263
Chapter 5	Passing Arguments to a Method, p. 279 Returning a Value from a Method, p. 293 The Retail Price Calculator Problem, p. 312
Chapter 6	Writing Classes and Creating Objects, p. 327 Initializing an Object with a Constructor, p. 348 The Personal Information Class Problem, p. 397
Chapter 7	Accessing Array Elements in a Loop, p. 409 Passing an Array to a Method, p. 424 The Charge Account Validation Problem, p. 489
Chapter 8	Returning Objects from Methods, p. 505 Aggregation, p. 517 The <code>BankAccount</code> , Class Copy Constructor Problem, p. 554
Chapter 9	The Sentence Capitalizer Problem, p. 608
Chapter 10	Inheritance, p. 613 Polymorphism, p. 657 The <code>Employee</code> and <code>Productionworker</code> Classes Problem, p. 698
Chapter 11	Handling Exceptions, p. 703 The Exception Project Problem, p. 759

(continued on the next page)

LOCATION OF VIDEONOTES IN THE TEXT *(continued)*



Chapter 12	Creating a Simple GUI Application, p. 764 Handling Events, p. 777 The Monthly Sales Tax Problem, p. 846
Chapter 13	The <code>JList</code> Component, p. 852 The <code>JComboBox</code> Component, p. 868 The Image Viewer Problem, p. 912
Chapter 14	Creating an Applet, p. 929 The <code>House</code> Applet Problem, p. 988
Chapter 15	Using Scene Builder to Create the Kilometer Converter GUI, p. 998 Learning More About the Main Application Class, p. 1010 Writing the Main Application Class For the Kilometer Converter GUI, p. 1011 Learning More About the Controller Class, p. 1013 Registering the Controller Class with the Application's GUI, p. 1014 JavaFX <code>RadioButtons</code> , p. 1019 JavaFX <code>CheckBoxes</code> , p. 1027 The Retail Price Calculator Problem, p. 1042
Chapter 16	Reducing a Problem with Recursion, p. 1051 The Recursive Power Problem, p. 1073
Chapter 17	Displaying Query Results in a <code>JTable</code> , p. 1129 The Customer Inserter Problem, p. 1168

Preface

Welcome to *Starting Out with Java: From Control Structures through Objects*, Sixth Edition. This book is intended for a one-semester or a two-quarter CS1 course. Although it is written for students with no prior programming background, even experienced students will benefit from its depth of detail.

Control Structures First, Then Objects

This text first introduces the student to the fundamentals of data types, input and output, control structures, methods, and objects created from standard library classes.

Next, the student learns to use arrays of primitive types and reference types. After this, the student progresses through more advanced topics, such as inheritance, polymorphism, the creation and management of packages, GUI applications, recursion, and database programming. From early in the book, applications are documented with javadoc comments. As the student progresses through the text, new javadoc tags are covered and demonstrated.

As with all the books in the *Starting Out With . . .* series, the hallmark of this text is its clear, friendly, and easy-to-understand writing. In addition, it is rich in example programs that are concise and practical.

Changes in This Edition

This book's pedagogy, organization, and clear writing style remain the same as in the previous edition. Many improvements have been made, which are summarized here:

- **A New Chapter on JavaFX:** New to this edition is *Chapter 15 Creating GUI Applications with JavaFX and Scene Builder*. JavaFX is the next generation toolkit for creating GUIs and graphical applications in Java, and is bundled with Java 8. This new chapter introduces the student to the JavaFX library, and shows how to use Scene Builder (a free download from Oracle) to visually design GUIs. The chapter is written in such a way that it is independent from the existing chapters on Swing and AWT. The instructor can choose to skip the Swing and AWT chapters and go straight to JavaFX, or cover all of the GUI chapters.

- **String.format Is Used Instead of DecimalFormat:** In previous editions, the `DecimalFormat` class was used to format strings for GUI output. In this edition, the `String.format` method is used instead. With `String.format`, the student can use the same format specifiers and flags that were learned with the `System.out.printf` method.
- **StringTokenizer Is No Longer Used:** In previous editions, the `StringTokenizer` class was introduced as a way to tokenize strings. In this edition, all string tokenizing is done with the `String.split` method.
- **Introduction of @Override annotation:** Chapter 10 now introduces the use of `@Override` annotation, and explains how it can prevent subtle errors.
- **A New Section on Anonymous Inner Classes:** Chapter 10 now has a new section that introduces anonymous inner classes.
- **The Introduction to Interfaces Has Been Improved:** The introductory material on interfaces in Chapter 10 has been revised for greater clarity.
- **Default Methods:** In this edition, Chapter 10 provides new material on default methods in interfaces, a new feature in Java 8.
- **Functional Interfaces and Lambda Expressions:** Java 8 introduces functional interfaces and lambda expressions, and in this edition, Chapter 10 has a new section on these topics. The new material gives a detailed, stepped-out explanation of lambda expressions, and discusses how they can be used to instantiate objects of anonymous classes that implement functional interfaces.
- **New Programming Problems:** Several new motivational programming problems have been added to many of the chapters.

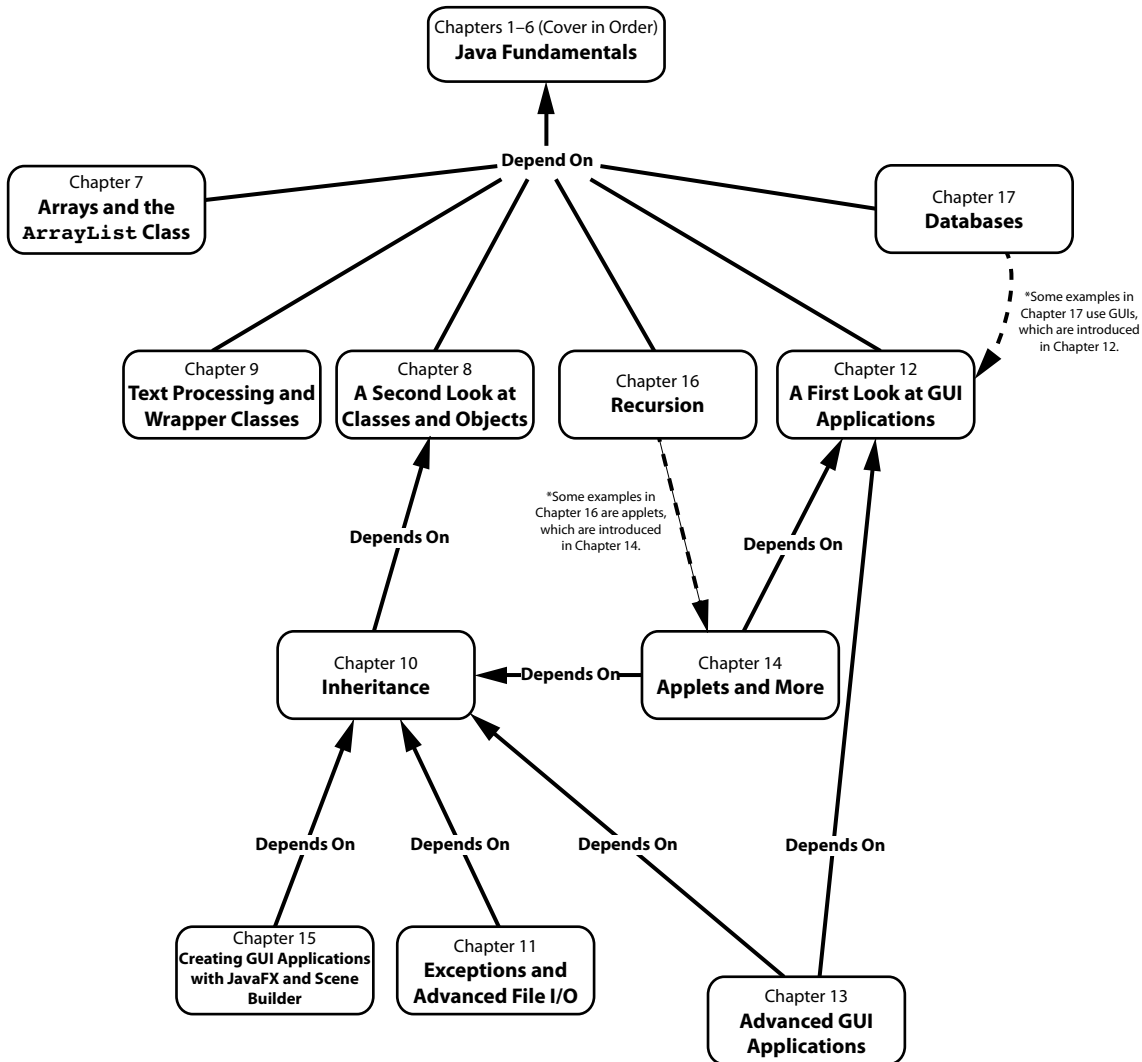
Organization of the Text

The text teaches Java step-by-step. Each chapter covers a major set of topics and builds knowledge as students progress through the book. Although the chapters can be easily taught in their existing sequence, there is some flexibility. Figure P-1 shows chapter dependencies. Each box represents a chapter or a group of chapters. An arrow points from a chapter to the chapter that must be previously covered.

Brief Overview of Each Chapter

Chapter 1: Introduction to Computers and Java. This chapter provides an introduction to the field of computer science and covers the fundamentals of hardware, software, and programming languages. The elements of a program, such as key words, variables, operators, and punctuation, are discussed by examining a simple program. An overview of entering source code, compiling, and executing a program is presented. A brief history of Java is also given.

Figure P-1 Chapter dependencies



Chapter 2: Java Fundamentals. This chapter gets students started in Java by introducing data types, identifiers, variable declarations, constants, comments, program output, and simple arithmetic operations. The conventions of programming style are also introduced. Students learn to read console input with the scanner class and with dialog boxes using `JOptionPane`.

Chapter 3: Decision Structures. In this chapter students explore relational operators and relational expressions and are shown how to control the flow of a program with the `if`, `if-else`, and `if-else-if` statements. Nested `if` statements, logical operators, the conditional operator, and the `switch` statement are also covered. The chapter discusses how to compare `String` objects with the `equals`, `compareTo`, `equalsIgnoreCase`, and `compareToIgnoreCase` methods. Formatting numeric output with the `System.out.printf` method and the `String.format` method is discussed.

Chapter 4: Loops and Files. This chapter covers Java’s repetition control structures. The `while` loop, `do-while` loop, and `for` loop are taught, along with common uses for these devices. Counters, accumulators, running totals, sentinels, and other application-related topics are discussed. Simple file operations for reading and writing text files are included.

Chapter 5: Methods. In this chapter students learn how to write `void` methods, value-returning methods, and methods that do and do not accept arguments. The concept of functional decomposition is discussed.

Chapter 6: A First Look at Classes. This chapter introduces students to designing classes for the purpose of instantiating objects. Students learn about class fields and methods, and UML diagrams are introduced as a design tool. Then constructors and overloading are discussed. A `BankAccount` class is presented as a case study, and a section on object-oriented design is included. This section leads the students through the process of identifying classes and their responsibilities within a problem domain. There is also a section that briefly explains packages and the `import` statement.

Chapter 7: Arrays and the ArrayList Class. In this chapter students learn to create and work with single and multi-dimensional arrays. Numerous array-processing techniques are demonstrated, such as summing the elements in an array, finding the highest and lowest values, and sequentially searching an array. Other topics, including ragged arrays and variable-length arguments (`varargs`), are also discussed. The `ArrayList` class is introduced, and Java’s generic types are briefly discussed and demonstrated.

Chapter 8: A Second Look at Classes and Objects. This chapter shows students how to write classes with added capabilities. Static methods and fields, interaction between objects, passing objects as arguments, and returning objects from methods are discussed. Aggregation and the “has a” relationship is covered, as well as enumerated types. A section on object-oriented design shows how to use CRC cards to determine the collaborations among classes.

Chapter 9: Text Processing and More about Wrapper Classes. This chapter discusses the numeric and `Character` wrapper classes. Methods for converting numbers to strings, testing the case of characters, and converting the case of characters are covered. Autoboxing and unboxing are also discussed. More `String` class methods are covered, including using the `split` method to tokenize strings. The chapter also covers the `StringBuilder` and `StringTokenizer` classes.

Chapter 10: Inheritance. The study of classes continues in this chapter with the subjects of inheritance and polymorphism. The topics covered include superclasses, subclasses, how constructors work in inheritance, method overriding, polymorphism and dynamic binding, protected and package access, class hierarchies, abstract classes, abstract methods, anonymous inner classes, interfaces, and lambda expressions.

Chapter 11: Exceptions and Advanced File I/O. In this chapter students learn to develop enhanced error trapping techniques using exceptions. Handling exceptions is covered, as well as developing and throwing custom exceptions. The chapter discusses advanced techniques for working with sequential access, random access, text, and binary files.

Chapter 12: A First Look at GUI Applications. This chapter presents the basics of developing GUI applications with Swing. Fundamental Swing components and the basic concepts of event-driven programming are covered.

Chapter 13: Advanced GUI Applications. This chapter continues the study of GUI application development with Swing. More advanced components, menu systems, and look-and-feel are covered.

Chapter 14: Applets and More. In this chapter students apply their knowledge of GUI development to the creation of applets. In addition to using Swing applet classes, AWT classes are discussed for portability. Drawing simple graphical shapes is discussed.

Chapter 15: Creating GUI Applications with JavaFX and Scene Builder. This chapter introduces JavaFX, which is the next generation library for creating graphical applications in Java. This chapter also shows how to use Scene Builder, a free screen designer from Oracle, to visually design GUIs. This chapter is written in such a way that it is independent from the existing chapters on Swing and AWT. You can choose to skip chapters 12, 13, and 14, and go straight to Chapter 15, or cover all of the GUI chapters.

Chapter 16: Recursion. This chapter presents recursion as a problem-solving technique. Numerous examples of recursive methods are demonstrated.

Chapter 17: Databases. This chapter introduces the student to database programming. The basic concepts of database management systems and SQL are first introduced. Then the student learns to use JDBC to write database applications in Java. Relational data is covered, and numerous example programs are presented throughout the chapter.

Features of the Text

Concept Statements. Each major section of the text starts with a concept statement that concisely summarizes the focus of the section.

Example Programs. The text has an abundant number of complete and partial example programs, each designed to highlight the current topic. In most cases the programs are practical, real-world examples.

Program Output. Each example program is followed by a sample of its output, which shows students how the program functions.



Checkpoints. Checkpoints, highlighted by the checkmark icon, appear at intervals throughout each chapter. They are designed to check students' knowledge soon after learning a new topic. Answers for all Checkpoint questions are provided in Appendix K, which can be downloaded from the book's resource page at www.pearsonhighered.com/cs-resources.



NOTE: Notes appear at several places throughout the text. They are short explanations of interesting or often misunderstood points relevant to the topic at hand.

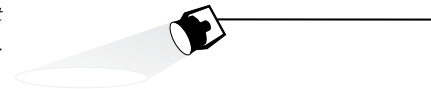


TIP: Tips advise the student on the best techniques for approaching different programming problems and appear regularly throughout the text.



WARNING! Warnings caution students about certain Java features, programming techniques, or practices that can lead to malfunctioning programs or lost data.

In the Spotlight. Many of the chapters provide an *In the Spotlight* section that presents a programming problem, along with detailed, step-by-step analysis showing the student how to solve it.



VideoNotes. A series of videos, developed specifically for this book, are available at www.pearsonhighered.com/gaddis. Icons appear throughout the text alerting the student to videos about specific topics.

Case Studies. Case studies that simulate real-world business applications are introduced throughout the text and are provided on the book's resource page at www.pearsonhighered.com/gaddis.

Common Errors to Avoid. Each chapter provides a list of common errors and explanations of how to avoid them.

Review Questions and Exercises. Each chapter presents a thorough and diverse set of review questions and exercises. They include Multiple Choice and True/False, Find the Error, Algorithm Workbench, and Short Answer.

Programming Challenges. Each chapter offers a pool of programming challenges designed to solidify students' knowledge of topics at hand. In most cases the assignments present real-world problems to be solved.

Supplements

Student Online Resources

Many student resources are available for this book from the publisher. The following items are available on the Gaddis Series resource page at www.pearsonhighered.com/cs-resources:

- The source code for each example program in the book
- Access to the book's companion VideoNotes
- Appendixes A–M (listed in the Contents)
- A collection of seven valuable Case Studies (listed in the Contents)
- Links to download the Java™ Edition Development Kit
- Links to download numerous programming environments including jGRASP™, Eclipse™, TextPad™, NetBeans™, JCreator, and DrJava

Online Practice and Assessment with MyProgrammingLab

MyProgrammingLab helps students fully grasp the logic, semantics, and syntax of programming. Through practice exercises and immediate, personalized feedback, MyProgrammingLab improves the programming competence of beginning students, who often struggle with the basic concepts and paradigms of popular high-level programming languages. A self-study and homework tool, the MyProgrammingLab course consists of hundreds of small practice exercises organized around the structure of this textbook. For students, the system automatically detects errors in the logic and syntax of their code submissions and offers targeted hints that enable students to figure out what went wrong—and why. For instructors, a comprehensive gradebook tracks correct and incorrect answers and stores the code inputted by students for review.

MyProgrammingLab is offered to users of this book in partnership with Turing's Craft, the makers of the CodeLab interactive programming exercise system. For a full demonstration, to see feedback from instructors and students, or to get started using MyProgrammingLab in your course, visit www.myprogramminglab.com.

Instructor Resources

The following supplements are available to qualified instructors:

- Answers to all of the Review Questions
- Solutions for the Programming Challenges
- PowerPoint Presentation slides for each chapter
- Computerized Test Banks
- Source Code
- Lab Manual
- Student Files for the Lab Manual
- Solutions to the Lab Manual

Visit the Pearson Instructor Resource Center (www.pearsonhighered.com/irc) or contact your local Pearson representative for information on how to access these resources.

Acknowledgments

There have been many helping hands in the development and publication of this book. We would like to thank the following faculty reviewers for their helpful suggestions and expertise:

Reviewers For This Edition

Carl Stephen Guynes
University of North Texas

Alan G. Jackson
Oakland Community College

Zhen Jiang
West Chester University

Neven Jurkovic
Palo Alto College

Dennis Lang
Kansas State University

Jiang Li
Austin Peay State University

Cheng Luo
Coppin State University

Felix Rodriguez
Naugatuck Valley Community College

Diane Rudolph
John A Logan College

Timothy Urness
Drake University

Zijiang Yang
Western Michigan University

Reviewers of Previous Editions

Ahmad Abuhejleh
University of Wisconsin, River Falls

Colin Archibald
Valencia Community College

Ijaz Awani
Savannah State University

Bill Bane
Tarleton State University

N. Dwight Barnette
Virginia Tech

Asoke Bhattacharyya
Saint Xavier University, Chicago

Marvin Bishop
Manhattan College

Heather Booth
University of Tennessee, Knoxville

David Boyd
Valdosta University

Julius Brandstatter
Golden Gate University

Kim Cannon
Greenville Tech

Jesse Cecil
College of the Siskiyous

James Chegwidan
Tarrant County College

Kay Chen
Bucks County Community College

Brad Chilton
Tarleton State University

Diane Christie
University of Wisconsin, Stout

- Cara Cocking
Marquette University
- Jose Cordova
University of Louisiana, Monroe
- Walter C. Daugherity
Texas A & M University
- Michael Doherty
University of the Pacific
- Jeanne M. Douglas
University of Vermont
- Sander Eller
*California Polytechnic University,
Pomona*
- Brooke Estabrook-Fishinghawk
Mesa Community College
- Mike Fry
Lebanon Valley College
- David Goldschmidt
College of St. Rose
- Georgia R. Grant
College of San Mateo
- Nancy Harris
James Madison University
- Chris Haynes
Indiana University
- Ric Heishman
Northern Virginia Community College
- Deedee Herrera
Dodge City Community College
- Mary Hovik
Lehigh Carbon Community College
- Brian Howard
DePauw University
- Alan Jackson
Oakland Community College (MI)
- Norm Jacobson
University of California, Irvine
- Stephen Judd
University of Pennsylvania
- Harry Lichtbach
Evergreen Valley College
- Michael A. Long
California State University, Chico
- Tim Margush
University of Akron
- Blayne E. Mayfield
Oklahoma State University
- Scott McLeod
Riverside Community College
- Dean Mellas
Cerritos College
- Georges Merx
San Diego Mesa College
- Martin Meyers
California State University, Sacramento
- Pati Milligan
Baylor University
- Laurie Murphy
Pacific Lutheran University
- Steve Newberry
Tarleton State University
- Lynne O'Hanlon
Los Angeles Pierce College
- Merrill Parker
*Chattanooga State Technical
Community College*
- Bryson R. Payne
*North Georgia College and State
University*
- Rodney Pearson
Mississippi State University
- Peter John Polito
Springfield College
- Charles Robert Putnam
*California State University,
Northridge*
- Y. B. Reddy
Grambling State University

Elizabeth Riley
Macon State College

Carolyn Schauble
Colorado State University

Bonnie Smith
Fresno City College

Daniel Spiegel
Kutztown University

Caroline St. Clair
North Central College

Karen Stanton
Los Medanos College

Peter van der Goes
Rose State College

Tuan A Vo
Mt. San Antonio College

Xiaoying Wang
University of Mississippi

Yu Wu
University of North Texas

I also want to thank everyone at Pearson for making the *Starting Out With . . .* series so successful. I have worked so closely with the team at Pearson that I consider them among my closest friends. I am extremely fortunate to have Matt Goldstein as my editor, and Kelsey Loanes as Editorial Assistant. They have guided me through the process of revising this book, as well as many others. I am also fortunate to have Demetrius Hall and Bram Van Kempen as Marketing Managers. Their hard work is truly inspiring, and they do a great job getting my books out to the academic community. The production team, led by Camille Trentacoste, worked tirelessly to make this book a reality. Thanks to you all!

About the Author

Tony Gaddis is the principal author of the *Starting Out With . . .* series of textbooks. He has nearly two decades of experience teaching computer science courses, primarily at Haywood Community College. Tony is a highly acclaimed instructor who was previously selected as the North Carolina Community College “Teacher of the Year” and has received the Teaching Excellence award from the National Institute for Staff and Organizational Development. The *Starting Out With . . .* series includes introductory textbooks covering programming logic and design, C++, Java™, Microsoft® Visual Basic®, Microsoft® Visual C#, Python, Alice, and App Inventor, all published by Pearson.



get with the programming

Through the power of practice and immediate personalized feedback, MyProgrammingLab improves your performance.

MyProgrammingLab™

Learn more at www.myprogramminglab.com

This page intentionally left blank

STARTING OUT WITH

JAVA™



From Control Structures
through Objects

This page intentionally left blank

Introduction to Computers and Java

TOPICS

- | | |
|---|---------------------------------|
| 1.1 Introduction | 1.4 Programming Languages |
| 1.2 Why Program? | 1.5 What Is a Program Made Of? |
| 1.3 Computer Systems: Hardware and Software | 1.6 The Programming Process |
| | 1.7 Object-Oriented Programming |

1.1 Introduction

This book teaches programming using Java. Java is a powerful language that runs on practically every type of computer. It can be used to create large applications or small programs that are part of a Web site. Before plunging right into learning Java, however, this chapter will review the fundamentals of computer hardware and software, and then take a broad look at computer programming in general.

1.2 Why Program?

CONCEPT: Computers can do many different jobs because they are programmable.

Every profession has tools that make the job easier to do. Carpenters use hammers, saws, and measuring tapes. Mechanics use wrenches, screwdrivers, and ratchets. Electronics technicians use probes, scopes, and meters. Some tools are unique and can be categorized as belonging to a single profession. For example, surgeons have certain tools that are designed specifically for surgical operations. Those tools probably aren't used by anyone other than surgeons. There are some tools, however, that are used in several professions. Screwdrivers, for instance, are used by mechanics, carpenters, and many others.

The computer is a tool used by so many professions that it cannot be easily categorized. It can perform so many different jobs that it is perhaps the most versatile tool ever made. To the accountant, computers balance books, analyze profits and losses, and prepare tax reports. To the factory worker, computers control manufacturing machines and track production. To the mechanic, computers analyze the various systems in an automobile and pinpoint hard-to-find problems. The computer can do such a wide variety of tasks because it can

be *programmed*. It is a machine specifically designed to follow instructions. Because of the computer's programmability, it doesn't belong to any single profession. Computers are designed to do whatever job their programs, or *software*, tell them to do.

Computer programmers do a very important job. They create software that transforms computers into the specialized tools of many trades. Without programmers, the users of computers would have no software, and without software, computers would not be able to do anything.

Computer programming is both an art and a science. It is an art because every aspect of a program should be carefully designed. Here are a few of the things that must be designed for any real-world computer program:

- The logical flow of the instructions
- The mathematical procedures
- The layout of the programming statements
- The appearance of the screens
- The way information is presented to the user
- The program's "user friendliness"
- Manuals, help systems, and/or other forms of written documentation

There is also a science to programming. Because programs rarely work right the first time they are written, a lot of analyzing, experimenting, correcting, and redesigning is required. This demands patience and persistence of the programmer. Writing software demands discipline as well. Programmers must learn special languages such as Java because computers do not understand English or other human languages. Programming languages have strict rules that must be carefully followed.

Both the artistic and scientific nature of programming makes writing computer software like designing a car: Both cars and programs should be functional, efficient, powerful, easy to use, and pleasing to look at.

1.3

Computer Systems: Hardware and Software

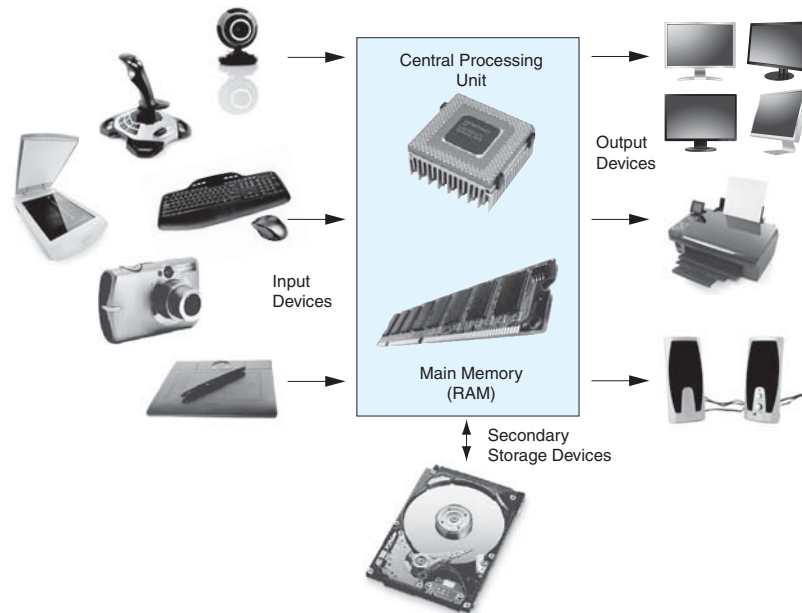
CONCEPT: All computer systems consist of similar hardware devices and software components.

Hardware

Hardware refers to the physical components that a computer is made of. A computer, as we generally think of it, is not an individual device, but a system of devices. Like the instruments in a symphony orchestra, each device plays its own part. A typical computer system consists of the following major components:

- The central processing unit (CPU)
- Main memory
- Secondary storage devices
- Input devices
- Output devices

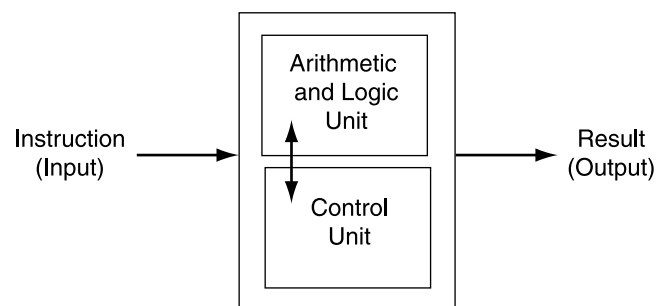
The organization of a computer system is shown in Figure 1-1.

Figure 1-1 The organization of a computer system

Let's take a closer look at each of these devices.

The CPU

At the heart of a computer is its *central processing unit*, or *CPU*. The CPU's job is to fetch instructions, follow the instructions, and produce some resulting data. Internally, the central processing unit consists of two parts: the *control unit* and the *arithmetic and logic unit (ALU)*. The control unit coordinates all of the computer's operations. It is responsible for determining where to get the next instruction and regulating the other major components of the computer with control signals. The arithmetic and logic unit, as its name suggests, is designed to perform mathematical operations. The organization of the CPU is shown in Figure 1-2.

Figure 1-2 The organization of the CPU

A program is a sequence of instructions stored in the computer's memory. When a computer is running a program, the CPU is engaged in a process known formally as the *fetch/decode/execute cycle*. The steps in the fetch/decode/execute cycle are as follows:

- Fetch* The CPU's control unit fetches, from main memory, the next instruction in the sequence of program instructions.
- Decode* The instruction is encoded in the form of a number. The control unit decodes the instruction and generates an electronic signal.
- Execute* The signal is routed to the appropriate component of the computer (such as the ALU, a disk drive, or some other device). The signal causes the component to perform an operation.

These steps are repeated as long as there are instructions to perform.

Main Memory

Commonly known as *random access memory*, or *RAM*, the computer's main memory is a device that holds information. Specifically, RAM holds the sequences of instructions in the programs that are running and the data those programs are using.

Memory is divided into sections that hold an equal amount of data. Each section is made of eight "switches" that may be either on or off. A switch in the on position usually represents the number 1, whereas a switch in the off position usually represents the number 0. The computer stores data by setting the switches in a memory location to a pattern that represents a character or a number. Each of these switches is known as a *bit*, which stands for *binary digit*. Each section of memory, which is a collection of eight bits, is known as a *byte*. Each byte is assigned a unique number known as an *address*. The addresses are ordered from lowest to highest. A byte is identified by its address in much the same way a post office box is identified by an address. Figure 1-3 shows a series of bytes with their addresses. In the illustration, sample data is stored in memory. The number 149 is stored in the byte at address 16, and the number 72 is stored in the byte at address 23.

RAM is usually a volatile type of memory, used only for temporary storage. When the computer is turned off, the contents of RAM are erased.

Figure 1-3 Memory bytes and their addresses

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16 149	17	18	19
20	21	22	23 72	24	25	26	27	28	29

Secondary Storage

Secondary storage is a type of memory that can hold data for long periods of time—even when there is no power to the computer. Frequently used programs are stored in secondary memory and loaded into main memory as needed. Important data, such as word processing documents, payroll data, and inventory figures, is saved to secondary storage as well.

The most common type of secondary storage device is the *disk drive*. A traditional disk drive stores data by magnetically encoding it onto a spinning circular disk. *Solid state drives*, which store data in solid-state memory, are increasingly becoming popular. A solid-state drive has no moving parts, and operates faster than a traditional disk drive.

Most computers have some sort of secondary storage device, either a traditional disk drive or a solid-state drive, mounted inside their case. External drives are also available, which connect to one of the computer's communication ports. External drives can be used to create backup copies of important data or to move data to another computer.

In addition to external drives, many types of devices have been created for copying data, and for moving it to other computers. *Universal Serial Bus drives*, or *USB drives* are small devices that plug into the computer's USB (Universal Serial Bus) port, and appear to the system as a disk drive. These drives do not actually contain a disk, however. They store data in a special type of memory known as *flash memory*. USB drives are inexpensive, reliable, and small enough to be carried in your pocket.

Optical devices such as the *CD* (compact disc) and the *DVD* (digital versatile disc) are also popular for data storage. Data is not recorded magnetically on an optical disc, but is encoded as a series of pits on the disc surface. CD and DVD drives use a laser to detect the pits and thus read the encoded data. Optical discs hold large amounts of data, and because recordable CD and DVD drives are now commonplace, they make a good medium for creating backup copies of data.

Input Devices

Input is any data the computer collects from the outside world. The device that collects the data and sends it to the computer is called an *input device*. Common input devices are the keyboard, mouse, scanner, and digital camera. Disk drives, optical drives, and USB drives can also be considered input devices because programs and data are retrieved from them and loaded into the computer's memory.

Output Devices

Output is any data the computer sends to the outside world. It might be a sales report, a list of names, or a graphic image. The data is sent to an output device, which formats and presents it. Common output devices are monitors and printers. Disk drives, USB drives, and CD recorders can also be considered output devices because the CPU sends data to them to be saved.

Software

As previously mentioned, software refers to the programs that run on a computer. There are two general categories of software: operating systems and application software. An operating system is a set of programs that manages the computer's hardware devices and controls their processes. Most all modern operating systems are multitasking, which means they are capable of running multiple programs at once. Through a technique called time sharing, a multitasking system divides the allocation of hardware resources and the attention of the CPU among all the executing programs. UNIX, Linux, Mac OS, and Windows are multitasking operating systems.

Application software refers to programs that make the computer useful to the user. These programs solve specific problems or perform general operations that satisfy the needs of the user. Word processing, spreadsheet, and database packages are all examples of application software.

**Checkpoint**MyProgrammingLab™ www.myprogramminglab.com

- 1.1 Why is the computer used by so many different people, in so many different professions?
- 1.2 List the five major hardware components of a computer system.
- 1.3 Internally, the CPU consists of what two units?
- 1.4 Describe the steps in the fetch/decode/execute cycle.
- 1.5 What is a memory address? What is its purpose?
- 1.6 Explain why computers have both main memory and secondary storage.
- 1.7 What does the term *multitasking* mean?

1.4 Programming Languages

CONCEPT: A program is a set of instructions a computer follows in order to perform a task. A programming language is a special language used to write computer programs.

What Is a Program?

Computers are designed to follow instructions. A computer program is a set of instructions that enable the computer to solve a problem or perform a task. For example, suppose we want the computer to calculate someone's gross pay. The following is a list of things the computer should do to perform this task.

1. Display a message on the screen: "How many hours did you work?"
2. Allow the user to enter the number of hours worked.
3. Once the user enters a number, store it in memory.
4. Display a message on the screen: "How much do you get paid per hour?"
5. Allow the user to enter an hourly pay rate.
6. Once the user enters a number, store it in memory.
7. Once both the number of hours worked and the hourly pay rate are entered, multiply the two numbers and store the result in memory.
8. Display a message on the screen that shows the amount of money earned. The message must include the result of the calculation performed in Step 7.

Collectively, these instructions are called an *algorithm*. An algorithm is a set of well-defined steps for performing a task or solving a problem. Notice that these steps are sequentially ordered. Step 1 should be performed before Step 2, and so forth. It is important that these instructions be performed in their proper sequence.

Although you and I might easily understand the instructions in the pay-calculating algorithm, it is not ready to be executed on a computer. A computer's CPU can only process instructions that are written in *machine language*. If you were to look at a machine language program, you would see a stream of binary numbers (numbers consisting of only 1s and 0s). The binary numbers form machine language instructions, which the CPU interprets as commands. Here is an example of what a machine language instruction might look like:

```
101101000000101
```

As you can imagine, the process of encoding an algorithm in machine language is very tedious and difficult. In addition, each different type of CPU has its own machine language. If you wrote a machine language program for computer A and then wanted to run it on computer B, which has a different type of CPU, you would have to rewrite the program in computer B's machine language.

Programming languages, which use words instead of numbers, were invented to ease the task of programming. A program can be written in a programming language, which is much easier to understand than machine language, and then translated into machine language. Programmers use software to perform this translation. Many programming languages have been created. Table 1-1 lists a few of the well-known ones.

Table 1-1 Programming languages

Language	Description
BASIC	Beginners All-purpose Symbolic Instruction Code is a general-purpose, procedural programming language. It was originally designed to be simple enough for beginners to learn.
FORTRAN	FORmula TRANslator is a procedural language designed for programming complex mathematical algorithms.
COBOL	Common Business-Oriented Language is a procedural language designed for business applications.
Pascal	Pascal is a structured, general-purpose, procedural language designed primarily for teaching programming.
C	C is a structured, general-purpose, procedural language developed at Bell Laboratories.
C++	Based on the C language, C++ offers object-oriented features not found in C. C++ was also invented at Bell Laboratories.
C#	Pronounced "C sharp." It is a language invented by Microsoft for developing applications based on the Microsoft .NET platform.
Java	Java is an object-oriented language invented at Sun Microsystems, and is now owned by Oracle. It may be used to develop stand-alone applications that operate on a single computer, applications that run over the Internet from a Web server, and applets that run in a Web browser.
JavaScript	JavaScript is a programming language that can be used in a Web site to perform simple operations. Despite its name, JavaScript is not related to Java.
Perl	A general-purpose programming language used widely on Internet servers.
PHP	A programming language used primarily for developing Web server applications and dynamic Web pages.
Python	Python is an object-oriented programming language used in both business and academia. Many popular Web sites contain features developed in Python.
Ruby	Ruby is a simple but powerful object-oriented programming language. It can be used for a variety of purposes, from small utility programs to large Web applications.
Visual Basic	Visual Basic is a Microsoft programming language and software development environment that allows programmers to create Windows-based applications quickly.

A History of Java

In 1991 a team was formed at Sun Microsystems (a company that is now owned by Oracle) to speculate about the important technological trends that might emerge in the near future. The team, which was named the Green Team, concluded that computers would merge with consumer appliances. Their first project was to develop a handheld device named *7 (pronounced star seven) that could be used to control a variety of home entertainment devices. For the unit to work, it had to use a programming language that could be processed by all the devices it controlled. This presented a problem because different brands of consumer devices use different processors, each with its own machine language.

Because no such universal language existed, James Gosling, the team's lead engineer, created one. Programs written in this language, which was originally named Oak, were not translated into the machine language of a specific processor, but were translated into an intermediate language known as *byte code*. Another program would then translate the byte code into machine language that could be executed by the processor in a specific consumer device.

Unfortunately, the technology developed by the Green Team was ahead of its time. No customers could be found, mostly because the computer-controlled consumer appliance industry was just beginning. But rather than abandoning their hard work and moving on to other projects, the team saw another opportunity: the Internet. The Internet is a perfect environment for a universal programming language such as Oak. It consists of numerous different computer platforms connected together in a single network.

To demonstrate the effectiveness of its language, which was renamed Java, the team used it to develop a Web browser. The browser, named HotJava, was able to download and run small Java programs known as applets. This gave the browser the capability to display animation and interact with the user. HotJava was demonstrated at the 1995 SunWorld conference before a wowed audience. Later the announcement was made that Netscape would incorporate Java technology into its Navigator browser. Other Internet companies rapidly followed, increasing the acceptance and the influence of the Java language. Today, Java is very popular for developing not only applets for the Internet but also stand-alone applications.

Java Applications and Applets

There are two types of programs that may be created with Java: applications and applets. An application is a stand-alone program that runs on your computer. You have probably used several applications already, such as word processors, spreadsheets, database managers, and graphics programs. Although Java may be used to write these types of applications, other languages such as C, C++, and Visual Basic are also used.

In the previous section you learned that Java may also be used to create applets. The term *applet* refers to a small application, in the same way that the term *piglet* refers to a small pig. Unlike applications, an applet is designed to be transmitted over the Internet from a Web server, and then executed in a Web browser. Applets are important because they can be used to extend the capabilities of a Web page significantly.

Web pages are normally written in Hypertext Markup Language (HTML). HTML is limited, however, because it merely describes the content and layout of a Web page. HTML does not have sophisticated abilities such as performing math calculations and interacting with the user. A Web designer can write a Java applet to perform operations that are

normally performed by an application and embed it in a Web site. When someone visits the Web site, the applet is downloaded to the visitor's browser and executed.

Security

Any time content is downloaded from a Web server to a visitor's computer, security is an important concern. Because Java is a full-featured programming language, at first you might be suspicious of any Web site that transmits an applet to your computer. After all, couldn't a Java applet do harmful things, such as deleting the contents of the disk drive or transmitting private information to another computer? Fortunately, the answer is no. Web browsers run Java applets in a secure environment within your computer's memory and do not allow them to access resources, such as a disk drive, that are outside that environment.

1.5 What Is a Program Made Of?

CONCEPT: There are certain elements that are common to all programming languages.

Language Elements

All programming languages have some things in common. Table 1-2 lists the common elements you will find in almost every language.

Table 1-2 The common elements of a programming language

Language Element	Description
Key Words	These are words that have a special meaning in the programming language. They may be used for their intended purpose only. Key words are also known as <i>reserved words</i> .
Operators	Operators are symbols or words that perform operations on one or more operands. An operand is usually an item of data, such as a number.
Punctuation	Most programming languages require the use of punctuation characters. These characters serve specific purposes, such as marking the beginning or ending of a statement, or separating items in a list.
Programmer-Defined Names	Unlike key words, which are part of the programming language, these are words or names that are defined by the programmer. They are used to identify storage locations in memory and parts of the program that are created by the programmer. Programmer-defined names are often called <i>identifiers</i> .
Syntax	These are rules that must be followed when writing a program. Syntax dictates how key words and operators may be used, and where punctuation symbols must appear.