



DEITEL® DEVELOPER SERIES

Java® 9

for Programmers

Interactive Java *with*
JShell

**Java Platform
Module System**
(Project Jigsaw)



PAUL DEITEL • HARVEY DEITEL

DEITEL® DEVELOPER SERIES

The DEITEL® DEVELOPER SERIES is designed for professional programmers. The series presents focused treatments on emerging and mature technologies, including Java®, C++, C, C# and .NET, JavaScript®, Internet and web development, Android™ app development, iOS® app development, Swift™ and more. Each book in the series contains the same live-code teaching methodology used in the Deitels' HOW TO PROGRAM SERIES college textbooks—in this book, most concepts are presented in the context of completely coded, live apps.

ABOUT THE COVER

The cover art we selected reflects some key themes of *Java® 9 for Programmers*: The art includes a colorful collection of sea shells—one of Java 9's key features is **JShell**, which is **Java 9's REPL (read-eval-print-loop) for interactive Java**. JShell is one of the most important pedagogic and productivity enhancements to Java since it was announced in 1995. We discuss JShell in detail in Chapter 23 and show how you can use it for discovery and experimentation, rapid prototyping of code segments and to learn Java faster. The shell art is overlaid onto a sphere of jigsaw puzzle pieces representing the earth. The most important new software-engineering capability in Java 9 is the **Java Platform Module System** (Chapter 27)—which was developed by Project Jigsaw.



Cover graphic: Jigsaw Puzzles, © StacieStauffSmith Photos/Shutterstock
Sea Shells, © Mopic/Shutterstock



DEITEL & ASSOCIATES, INC.

Deitel & Associates, Inc., founded by Paul Deitel and Harvey Deitel, is an internationally recognized authoring and corporate training organization, specializing in computer programming languages, object technology, Internet and web software technology, and Android and iOS app development. The company's clients over the years have included many of the world's largest corporations, government agencies, branches of the military and academic institutions. The company offers instructor-led training courses delivered at client sites worldwide on major programming languages and platforms. Through its 42-year publishing partnership with Prentice Hall/Pearson, Deitel & Associates, Inc., creates leading-edge programming professional books, college textbooks, LiveLessons video products, Learning Paths in the Safari service (<http://www.safaribooksonline.com>), e-books and REVEL™ interactive multimedia courses with integrated labs and assessment (revel.pearson.com). To learn more about Deitel & Associates, Inc., its text and video publications and its worldwide instructor-led, on-site training curriculum, visit www.deitel.com or send an email to deitel@deitel.com. Join the Deitel social networking communities on LinkedIn® (bit.ly/DeitelLinkedIn), Facebook® (www.facebook.com/DeitelFan), Twitter® (twitter.com/deitel), and YouTube™ (www.youtube.com/DeitelTV), and subscribe to the *Deitel® Buzz Online* newsletter (www.deitel.com/newsletter/subscribe.html).

Java 9 for Programmers

Table of Contents

Cover

Half Title

Title Page

Copyright Page

Acknowledgments

Contents

Foreword

Preface

Before You Begin

1 Introduction and Test-Driving a Java Application

1.1 Introduction

1.2 Object Technology Concepts

1.2.1 Automobile as an Object

1.2.2 Methods and Classes

1.2.3 Instantiation

1.2.4 Reuse

1.2.5 Messages and Method Calls

1.2.6 Attributes and Instance Variables

1.2.7 Encapsulation and Information Hiding

1.2.8 Inheritance

1.2.9 Interfaces

1.2.10 Object-Oriented Analysis and Design (OOAD)

1.2.11 The UML (Unified Modeling Language)

Table of Contents

1.3 Java

1.4 A Typical Java Development Environment

1.5 Test-Driving a Java Application

1.6 Software Technologies

1.7 Getting Your Questions Answered

2 Introduction to Java Applications; Input/Output and Operators

2.1 Introduction

2.2 Your First Program in Java: Printing a Line of Text

2.2.1 Compiling the Application

2.2.2 Executing the Application

2.3 Modifying Your First Java Program

2.4 Displaying Text with printf

2.5 Another Application: Adding Integers

2.5.1 import Declarations

2.5.2 Declaring and Creating a Scanner to Obtain User Input from the
Keyboard

2.5.3 Prompting the User for Input

2.5.4 Declaring a Variable to Store an Integer and Obtaining an Integer from the
Keyboard

2.5.5 Obtaining a Second Integer

2.5.6 Using Variables in a Calculation

2.5.7 Displaying the Calculation Result

2.5.8 Java API Documentation

2.5.9 Declaring and Initializing Variables in Separate Statements

2.6 Arithmetic

2.7 Decision Making: Equality and Relational Operators

2.8 Wrap-Up

3 Introduction to Classes, Objects, Methods and Strings

Table of Contents

3.1 Introduction

3.2 Instance Variables, set Methods and get Methods

3.2.1 Account Class with an Instance Variable, and set and get Methods

3.2.2 AccountTest Class That Creates and Uses an Object of Class Account

3.2.3 Compiling and Executing an App with Multiple Classes

3.2.4 Account UML Class Diagram

3.2.5 Additional Notes on Class AccountTest

3.2.6 Software Engineering with private Instance Variables and public
set and get Methods

3.3 Account Class: Initializing Objects with Constructors

3.3.1 Declaring an Account Constructor for Custom Object Initialization

3.3.2 Class AccountTest: Initializing Account Objects When Theyre
Created

3.4 Account Class with a Balance; Floating-Point Numbers

3.4.1 Account Class with a balance Instance Variable of Type double

3.4.2 AccountTest Class to Use Class Account

3.5 Primitive Types vs. Reference Types

3.6 Wrap-Up

4 Control Statements: Part 1; Assignment, ++ and -- Operators

4.1 Introduction

4.2 Control Structures

4.2.1 Sequence Structure in Java

4.2.2 Selection Statements in Java

4.2.3 Iteration Statements in Java

4.2.4 Summary of Control Statements in Java

4.3 if Single-Selection Statement

4.4 ifelse Double-Selection Statement

4.4.1 Nested ifelse Statements

4.4.2 Dangling-else Problem

Table of Contents

4.4.3 Blocks

4.4.4 Conditional Operator (?:)

4.5 while Iteration Statement

4.6 Counter-Controlled Iteration

4.7 Sentinel-Controlled Iteration

4.8 Nesting Different Control Statements

4.9 Compound Assignment Operators

4.10 Increment and Decrement Operators

4.11 Primitive Types

4.12 Wrap-Up

5 Control Statements: Part 2; Logical Operators

5.1 Introduction

5.2 Essentials of Counter-Controlled Iteration

5.3 for Iteration Statement

5.4 Examples Using the for Statement

5.4.1 Application: Summing the Even Integers from 2 to 20

5.4.2 Application: Compound-Interest Calculations

5.5 dowhile Iteration Statement

5.6 switch Multiple-Selection Statement

5.7 Class AutoPolicy: Strings in switch Statements

5.8 break and continue Statements

5.8.1 break Statement

5.8.2 continue Statement

5.9 Logical Operators

5.9.1 Conditional AND (&&) Operator

5.9.2 Conditional OR (||) Operator

5.9.3 Short-Circuit Evaluation of Complex Conditions

Table of Contents

5.9.4 Boolean Logical AND (&) and Boolean Logical Inclusive OR (|)
Operators

5.9.5 Boolean Logical Exclusive OR (^)

5.9.6 Logical Negation (!) Operator

5.9.7 Logical Operators Example

5.10 Wrap-Up

6 Methods: A Deeper Look

6.1 Introduction

6.2 Program Units in Java

6.3 static Methods, static Fields and Class Math

6.4 Methods with Multiple Parameters

6.5 Notes on Declaring and Using Methods

6.6 Argument Promotion and Casting

6.7 Java API Packages

6.8 Case Study: Secure Random-Number Generation

6.9 Case Study: A Game of Chance; Introducing enum Types

6.10 Scope of Declarations

6.11 Method Overloading

6.11.1 Declaring Overloaded Methods

6.11.2 Distinguishing Between Overloaded Methods

6.11.3 Return Types of Overloaded Methods

6.12 Wrap-Up

7 Arrays and ArrayLists

7.1 Introduction

7.2 Arrays

7.3 Declaring and Creating Arrays

7.4 Examples Using Arrays

Table of Contents

- 7.4.1 Creating and Initializing an Array
- 7.4.2 Using an Array Initializer
- 7.4.3 Calculating the Values to Store in an Array
- 7.4.4 Summing the Elements of an Array
- 7.4.5 Using Bar Charts to Display Array Data Graphically
- 7.4.6 Using the Elements of an Array as Counters
- 7.4.7 Using Arrays to Analyze Survey Results
- 7.5 Exception Handling: Processing the Incorrect Response
 - 7.5.1 The try Statement
 - 7.5.2 Executing the catch Block
 - 7.5.3 toString Method of the Exception Parameter
- 7.6 Case Study: Card Shuffling and Dealing Simulation
- 7.7 Enhanced for Statement
- 7.8 Passing Arrays to Methods
- 7.9 Pass-By-Value vs. Pass-By-Reference
- 7.10 Case Study: Class GradeBook Using an Array to Store Grades
- 7.11 Multidimensional Arrays
 - 7.11.1 Arrays of One-Dimensional Arrays
 - 7.11.2 Two-Dimensional Arrays with Rows of Different Lengths
 - 7.11.3 Creating Two-Dimensional Arrays with Array-Creation Expressions
 - 7.11.4 Two-Dimensional Array Example: Displaying Element Values
 - 7.11.5 Common Multidimensional-Array Manipulations Performed with for Statements
- 7.12 Case Study: Class GradeBook Using a Two-Dimensional Array
- 7.13 Variable-Length Argument Lists
- 7.14 Using Command-Line Arguments
- 7.15 Class Arrays
- 7.16 Introduction to Collections and Class ArrayList

Table of Contents

7.17 Wrap-Up

8 Classes and Objects: A Deeper Look

8.1 Introduction

8.2 Time Class Case Study

8.3 Controlling Access to Members

8.4 Referring to the Current Objects Members with the this Reference

8.5 Time Class Case Study: Overloaded Constructors

8.6 Default and No-Argument Constructors

8.7 Notes on Set and Get Methods

8.8 Composition

8.9 enum Types

8.10 Garbage Collection

8.11 static Class Members

8.12 static Import

8.13 final Instance Variables

8.14 Package Access

8.15 Using BigDecimal for Precise Monetary Calculations

8.16 JavaMoney API

8.17 Time Class Case Study: Creating Packages

8.18 Wrap-Up

9 Object-Oriented Programming: Inheritance

9.1 Introduction

9.2 Superclasses and Subclasses

9.3 protected Members

9.4 Relationship Between Superclasses and Subclasses

9.4.1 Creating and Using a CommissionEmployee Class

9.4.2 Creating and Using a BasePlusCommissionEmployee Class

Table of Contents

9.4.3 Creating a CommissionEmployeeBasePlusCommissionEmployee
Inheritance Hierarchy

9.4.4 CommissionEmployeeBasePlusCommissionEmployee Inheritance
Hierarchy Using protected Instance Variables

9.4.5 CommissionEmployeeBasePlusCommissionEmployee Inheritance
Hierarchy Using private Instance Variables

9.5 Constructors in Subclasses

9.6 Class Object

9.7 Designing with Composition vs. Inheritance

9.8 Wrap-Up

10 Object-Oriented Programming: Polymorphism and Interfaces

10.1 Introduction

10.2 Polymorphism Examples

10.3 Demonstrating Polymorphic Behavior

10.4 Abstract Classes and Methods

10.5 Case Study: Payroll System Using Polymorphism

10.5.1 Abstract Superclass Employee

10.5.2 Concrete Subclass SalariedEmployee

10.5.3 Concrete Subclass HourlyEmployee

10.5.4 Concrete Subclass CommissionEmployee

10.5.5 Indirect Concrete Subclass BasePlusCommissionEmployee

10.5.6 Polymorphic Processing, Operator instanceof and Downcasting

10.6 Allowed Assignments Between Superclass and Subclass
Variables

10.7 final Methods and Classes

10.8 A Deeper Explanation of Issues with Calling Methods from
Constructors

Table of Contents

10.9 Creating and Using Interfaces

- 10.9.1 Developing a Payable Hierarchy
- 10.9.2 Interface Payable
- 10.9.3 Class Invoice
- 10.9.4 Modifying Class Employee to Implement Interface Payable
- 10.9.5 Using Interface Payable to Process Invoices and Employees Polymorphically
- 10.9.6 Some Common Interfaces of the Java API

10.10 Java SE 8 Interface Enhancements

- 10.10.1 default Interface Methods
- 10.10.2 static Interface Methods
- 10.10.3 Functional Interfaces

10.11 Java SE 9 private Interface Methods

10.12 private Constructors

10.13 Program to an Interface, Not an Implementation

- 10.13.1 Implementation Inheritance Is Best for Small Numbers of Tightly Coupled Classes
- 10.13.2 Interface Inheritance Is Best for Flexibility
- 10.13.3 Rethinking the Employee Hierarchy

10.14 Wrap-Up

11 Exception Handling: A Deeper Look

- 11.1 Introduction
- 11.2 Example: Divide by Zero without Exception Handling
- 11.3 Example: Handling ArithmeticExceptions and InputMismatchExceptions
- 11.4 When to Use Exception Handling
- 11.5 Java Exception Hierarchy
- 11.6 finally Block

Table of Contents

- 11.7 Stack Unwinding and Obtaining Information from an Exception
- 11.8 Chained Exceptions
- 11.9 Declaring New Exception Types
- 11.10 Preconditions and Postconditions
- 11.11 Assertions
- 11.12 try-with-Resources: Automatic Resource Deallocation
- 11.13 Wrap-Up

12 JavaFX Graphical User Interfaces: Part 1

- 12.1 Introduction
- 12.2 JavaFX Scene Builder
- 12.3 JavaFX App Window Structure
- 12.4 Welcome AppDisplaying Text and an Image
 - 12.4.1 Opening Scene Builder and Creating the File Welcome.fxml
 - 12.4.2 Adding an Image to the Folder Containing Welcome.fxml
 - 12.4.3 Creating a VBox Layout Container
 - 12.4.4 Configuring the VBox Layout Container
 - 12.4.5 Adding and Configuring a Label
 - 12.4.6 Adding and Configuring an ImageView
 - 12.4.7 Previewing the Welcome GUI
- 12.5 Tip Calculator AppIntroduction to Event Handling
 - 12.5.1 Test-Driving the Tip Calculator App
 - 12.5.2 Technologies Overview
 - 12.5.3 Building the Apps GUI
 - 12.5.4 TipCalculator Class
 - 12.5.5 TipCalculatorController Class
- 12.6 Features Covered in the Other JavaFX Chapters
- 12.7 Wrap-Up

Table of Contents

13 JavaFX GUI: Part 2

13.1 Introduction

13.2 Laying Out Nodes in a Scene Graph

13.3 Painter App: RadioButtons, Mouse Events and Shapes

13.3.1 Technologies Overview

13.3.2 Creating the Painter.fxml File

13.3.3 Building the GUI

13.3.4 Painter Subclass of Application

13.3.5 PainterController Class

13.4 Color Chooser App: Property Bindings and Property Listeners

13.4.1 Technologies Overview

13.4.2 Building the GUI

13.4.3 ColorChooser Subclass of Application

13.4.4 ColorChooserController Class

13.5 Cover Viewer App: Data-Driven GUIs with JavaFX Collections

13.5.1 Technologies Overview

13.5.2 Adding Images to the Apps Folder

13.5.3 Building the GUI

13.5.4 CoverViewer Subclass of Application

13.5.5 CoverViewerController Class

13.6 Cover Viewer App: Customizing ListView Cells

13.6.1 Technologies Overview

13.6.2 Copying the CoverViewer App

13.6.3 ImageTextCell Custom Cell Factory Class

13.6.4 CoverViewerController Class

13.7 Additional JavaFX Capabilities

13.8 JavaFX 9: Java SE 9 JavaFX Updates

Table of Contents

13.9 Wrap-Up

14 Strings, Characters and Regular Expressions

14.1 Introduction

14.2 Fundamentals of Characters and Strings

14.3 Class String

14.3.1 String Constructors

14.3.2 String Methods length, charAt and getChars

14.3.3 Comparing Strings

14.3.4 Locating Characters and Substrings in Strings

14.3.5 Extracting Substrings from Strings

14.3.6 Concatenating Strings

14.3.7 Miscellaneous String Methods

14.3.8 String Method valueOf

14.4 Class StringBuilder

14.4.1 StringBuilder Constructors

14.4.2 StringBuilder Methods length, capacity, setLength and ensureCapacity

14.4.3 StringBuilder Methods charAt, setCharAt, getChars and reverse

14.4.4 StringBuilder append Methods

14.4.5 StringBuilder Insertion and Deletion Methods

14.5 Class Character

14.6 Tokenizing Strings

14.7 Regular Expressions, Class Pattern and Class Matcher

14.7.1 Replacing Substrings and Splitting Strings

14.7.2 Classes Pattern and Matcher

14.8 Wrap-Up

15 Files, Input/Output Streams, NIO and XML Serialization

15.1 Introduction

Table of Contents

15.2 Files and Streams

15.3 Using NIO Classes and Interfaces to Get File and Directory Information

15.4 Sequential Text Files

15.4.1 Creating a Sequential Text File

15.4.2 Reading Data from a Sequential Text File

15.4.3 Case Study: A Credit-Inquiry Program

15.4.4 Updating Sequential Files

15.5 XML Serialization

15.5.1 Creating a Sequential File Using XML Serialization

15.5.2 Reading and Deserializing Data from a Sequential File

15.6 FileChooser and DirectoryChooser Dialogs

15.7 (Optional) Additional java.io Classes

15.7.1 Interfaces and Classes for Byte-Based Input and Output

15.7.2 Interfaces and Classes for Character-Based Input and Output

15.8 Wrap-Up

16 Generic Collections

16.1 Introduction

16.2 Collections Overview

16.3 Type-Wrapper Classes

16.4 Autoboxing and Auto-Unboxing

16.5 Interface Collection and Class Collections

16.6 Lists

16.6.1 ArrayList and Iterator

16.6.2 LinkedList

16.7 Collections Methods

16.7.1 Method sort

16.7.2 Method shuffle

Table of Contents

16.7.3 Methods reverse, fill, copy, max and min

16.7.4 Method binarySearch

16.7.5 Methods addAll, frequency and disjoint

16.8 Class PriorityQueue and Interface Queue

16.9 Sets

16.10 Maps

16.11 Synchronized Collections

16.12 Unmodifiable Collections

16.13 Abstract Implementations

16.14 Java SE 9: Convenience Factory Methods for Immutable Collections

16.15 Wrap-Up

17 Lambdas and Streams

17.1 Introduction

17.2 Streams and Reduction

17.2.1 Summing the Integers from 1 through 10 with a for Loop

17.2.2 External Iteration with for Is Error Prone

17.2.3 Summing with a Stream and Reduction

17.2.4 Internal Iteration

17.3 Mapping and Lambdas

17.3.1 Lambda Expressions

17.3.2 Lambda Syntax

17.3.3 Intermediate and Terminal Operations

17.4 Filtering

17.5 How Elements Move Through Stream Pipelines

17.6 Method References

17.6.1 Creating an IntStream of Random Values

17.6.2 Performing a Task on Each Stream Element with forEach and a

Table of Contents

Method Reference

17.6.3 Mapping Integers to String Objects with mapToObj

17.6.4 Concatenating Strings with collect

17.7 IntStream Operations

17.7.1 Creating an IntStream and Displaying Its Values

17.7.2 Terminal Operations count, min, max, sum and average

17.7.3 Terminal Operation reduce

17.7.4 Sorting IntStream Values

17.8 Functional Interfaces

17.9 Lambdas: A Deeper Look

17.10 Stream<Integer> Manipulations

17.10.1 Creating a Stream<Integer>

17.10.2 Sorting a Stream and Collecting the Results

17.10.3 Filtering a Stream and Storing the Results for Later Use

17.10.4 Filtering and Sorting a Stream and Collecting the Results

17.10.5 Sorting Previously Collected Results

17.11 Stream<String> Manipulations

17.11.1 Mapping Strings to Uppercase

17.11.2 Filtering Strings Then Sorting Them in Case-Insensitive
Ascending Order

17.11.3 Filtering Strings Then Sorting Them in Case-Insensitive
Descending Order

17.12 Stream<Employee> Manipulations

17.12.1 Creating and Displaying a List<Employee>

17.12.2 Filtering Employees with Salaries in a Specified Range

17.12.3 Sorting Employees By Multiple Fields

17.12.4 Mapping Employees to Unique-Last-Name Strings

17.12.5 Grouping Employees By Department

17.12.6 Counting the Number of Employees in Each Department

Table of Contents

17.12.7 Summing and Averaging Employee Salaries

17.13 Creating a Stream<String> from a File

17.14 Streams of Random Values

17.15 Infinite Streams

17.16 Lambda Event Handlers

17.17 Additional Notes on Java SE 8 Interfaces

17.18 Wrap-Up

18 Recursion

18.1 Introduction

18.2 Recursion Concepts

18.3 Example Using Recursion: Factorials

18.4 Reimplementing Class FactorialCalculator Using BigInteger

18.5 Example Using Recursion: Fibonacci Series

18.6 Recursion and the Method-Call Stack

18.7 Recursion vs. Iteration

18.8 Towers of Hanoi

18.9 Fractals

18.9.1 Koch Curve Fractal

18.9.2 (Optional) Case Study: Lo Feather Fractal

18.9.3 (Optional) Fractal App GUI

18.9.4 (Optional) FractalController Class

18.10 Recursive Backtracking

18.11 Wrap-Up

19 Generic Classes and Methods: A Deeper Look

19.1 Introduction

19.2 Motivation for Generic Methods

19.3 Generic Methods: Implementation and Compile-Time

Table of Contents

Translation

19.4 Additional Compile-Time Translation Issues: Methods That Use a Type Parameter as the Return Type

19.5 Overloading Generic Methods

19.6 Generic Classes

19.7 Wildcards in Methods That Accept Type Parameters

19.8 Wrap-Up

20 JavaFX Graphics, Animation and Video

20.1 Introduction

20.2 Controlling Fonts with Cascading Style Sheets (CSS)

20.2.1 CSS That Styles the GUI

20.2.2 FXML That Defines the GUI Introduction to XML Markup

20.2.3 Referencing the CSS File from FXML

20.2.4 Specifying the VBoxs Style Class

20.2.5 Programmatically Loading CSS

20.3 Displaying Two-Dimensional Shapes

20.3.1 Defining Two-Dimensional Shapes with FXML

20.3.2 CSS That Styles the Two-Dimensional Shapes

20.4 Polylines, Polygons and Paths

20.4.1 GUI and CSS

20.4.2 PolyShapesController Class

20.5 Transforms

20.6 Playing Video with Media, MediaPlayer and MediaPlayer

20.6.1 VideoPlayer GUI

20.6.2 VideoPlayerController Class

20.7 Transition Animations

20.7.1 TransitionAnimations.fxml

20.7.2 TransitionAnimationsController Class

Table of Contents

20.8 Timeline Animations

20.9 Frame-by-Frame Animation with AnimationTimer

20.10 Drawing on a Canvas

20.11 Three-Dimensional Shapes

20.12 Wrap-Up

21 Concurrency and Multi-Core Performance

21.1 Introduction

21.2 Thread States and Life Cycle

21.2.1 New and Runnable States

21.2.2 Waiting State

21.2.3 Timed Waiting State

21.2.4 Blocked State

21.2.5 Terminated State

21.2.6 Operating-System View of the Runnable State

21.2.7 Thread Priorities and Thread Scheduling

21.2.8 Indefinite Postponement and Deadlock

21.3 Creating and Executing Threads with the Executor Framework

21.4 Thread Synchronization

21.4.1 Immutable Data

21.4.2 Monitors

21.4.3 Unsynchronized Mutable Data Sharing

21.4.4 Synchronized Mutable Data Sharing Making Operations Atomic

21.5 Producer/Consumer Relationship without Synchronization

21.6 Producer/Consumer Relationship: ArrayBlockingQueue

21.7 (Advanced) Producer/Consumer Relationship with synchronized, wait, notify and notifyAll

21.8 (Advanced) Producer/Consumer Relationship: Bounded Buffers

21.9 (Advanced) Producer/Consumer Relationship: The Lock and

Table of Contents

Condition Interfaces

21.10 Concurrent Collections

21.11 Multithreading in JavaFX

21.11.1 Performing Computations in a Worker Thread: Fibonacci Numbers

21.11.2 Processing Intermediate Results: Sieve of Eratosthenes

21.12 sort/parallelSort Timings with the Java SE 8 Date/Time API

21.13 Java SE 8: Sequential vs. Parallel Streams

21.14 (Advanced) Interfaces Callable and Future

21.15 (Advanced) Fork/Join Framework

21.16 Wrap-Up

22 Accessing Databases with JDBC

22.1 Introduction

22.2 Relational Databases

22.3 A books Database

22.4 SQL

22.4.1 Basic SELECT Query

22.4.2 WHERE Clause

22.4.3 ORDER BY Clause

22.4.4 Merging Data from Multiple Tables: INNER JOIN

22.4.5 INSERT Statement

22.4.6 UPDATE Statement

22.4.7 DELETE Statement

22.5 Setting Up a Java DB Database

22.5.1 Creating the Chapters Databases on Windows

22.5.2 Creating the Chapters Databases on macOS

22.5.3 Creating the Chapters Databases on Linux

22.6 Connecting to and Querying a Database

22.6.1 Automatic Driver Discovery

Table of Contents

22.6.2 Connecting to the Database

22.6.3 Creating a Statement for Executing Queries

22.6.4 Executing a Query

22.6.5 Processing a Query's ResultSet

22.7 Querying the books Database

22.7.1 ResultSetTableModel Class

22.7.2 DisplayQueryResults App's GUI

22.7.3 DisplayQueryResultsController Class

22.8 RowSet Interface

22.9 PreparedStatements

22.9.1 AddressBook App That Uses PreparedStatements

22.9.2 Class Person

22.9.3 Class PersonQueries

22.9.4 AddressBook GUI

22.9.5 Class AddressBookController

22.10 Stored Procedures

22.11 Transaction Processing

22.12 Wrap-Up

23 Introduction to JShell: Java 9's REPL for Interactive Java

23.1 Introduction

23.2 Installing JDK 9

23.3 Introduction to JShell

23.3.1 Starting a JShell Session

23.3.2 Executing Statements

23.3.3 Declaring Variables Explicitly

23.3.4 Listing and Executing Prior Snippets

23.3.5 Evaluating Expressions and Declaring Variables Implicitly

23.3.6 Using Implicitly Declared Variables

Table of Contents

- 23.3.7 Viewing a Variables Value
- 23.3.8 Resetting a JShell Session
- 23.3.9 Writing Multiline Statements
- 23.3.10 Editing Code Snippets
- 23.3.11 Exiting JShell
- 23.4 Command-Line Input in JShell
- 23.5 Declaring and Using Classes
 - 23.5.1 Creating a Class in JShell
 - 23.5.2 Explicitly Declaring Reference-Type Variables
 - 23.5.3 Creating Objects
 - 23.5.4 Manipulating Objects
 - 23.5.5 Creating a Meaningful Variable Name for an Expression
 - 23.5.6 Saving and Opening Code-Snippet Files
- 23.6 Discovery with JShell Auto-Completion
 - 23.6.1 Auto-Completing Identifiers
 - 23.6.2 Auto-Completing JShell Commands
- 23.7 Exploring a Classs Members and Viewing Documentation
 - 23.7.1 Listing Class Maths static Members
 - 23.7.2 Viewing a Methods Parameters
 - 23.7.3 Viewing a Methods Documentation
 - 23.7.4 Viewing a public Fields Documentation
 - 23.7.5 Viewing a Classs Documentation
 - 23.7.6 Viewing Method Overloads
 - 23.7.7 Exploring Members of a Specific Object
- 23.8 Declaring Methods
 - 23.8.1 Forward Referencing an Undeclared MethodDeclaring Method
displayCubes
 - 23.8.2 Declaring a Previously Undeclared Method
 - 23.8.3 Testing cube and Replacing Its Declaration

Table of Contents

23.8.4 Testing Updated Method cube and Method displayCubes

23.9 Exceptions

23.10 Importing Classes and Adding Packages to the CLASSPATH

23.11 Using an External Editor

23.12 Summary of JShell Commands

23.12.1 Getting Help in JShell

23.12.2 /edit Command: Additional Features

23.12.3 /reload Command

23.12.4 /drop Command

23.12.5 Feedback Modes

23.12.6 Other JShell Features Configurable with /set

23.13 Keyboard Shortcuts for Snippet Editing

23.14 How JShell Reinterprets Java for Interactive Use

23.15 IDE JShell Support

23.16 Wrap-Up

24 Java Persistence API (JPA)

24.1 Introduction

24.2 JPA Technology Overview

24.2.1 Generated Entity Classes

24.2.2 Relationships Between Tables in the Entity Classes

24.2.3 The javax.persistence Package

24.3 Querying a Database with JPA

24.3.1 Creating the Java DB Database

24.3.2 Populate the books Database with Sample Data

24.3.3 Creating the Java Project

24.3.4 Adding the JPA and Java DB Libraries

24.3.5 Creating the Persistence Unit for the books Database

24.3.6 Querying the Authors Table

Table of Contents

24.3.7 JPA Features of Autogenerated Class Authors

24.4 Named Queries; Accessing Data from Multiple Tables

24.4.1 Using a Named Query to Get the List of Authors, then Display the Authors with Their Titles

24.4.2 Using a Named Query to Get the List of Titles, then Display Each with Its Authors

24.5 Address Book: Using JPA and Transactions to Modify a Database

24.5.1 Transaction Processing

24.5.2 Creating the AddressBook Database, Project and Persistence Unit

24.5.3 Addresses Entity Class

24.5.4 AddressBookController Class

24.5.5 Other JPA Operations

24.6 Web Resources

24.7 Wrap-Up

25 ATM Case Study, Part 1: Object-Oriented Design with the UML

25.1 Case Study Introduction

25.2 Examining the Requirements Document

25.3 Identifying the Classes in a Requirements Document

25.4 Identifying Class Attributes

25.5 Identifying Objects States and Activities

25.6 Identifying Class Operations

25.7 Indicating Collaboration Among Objects

25.8 Wrap-Up

26 ATM Case Study Part 2: Implementing an Object-Oriented Design

26.1 Introduction

Table of Contents

26.2 Starting to Program the Classes of the ATM System

26.3 Incorporating Inheritance and Polymorphism into the ATM System

26.4 ATM Case Study Implementation

26.4.1 Class ATM

26.4.2 Class Screen

26.4.3 Class Keypad

26.4.4 Class CashDispenser

26.4.5 Class DepositSlot

26.4.6 Class Account

26.4.7 Class BankDatabase

26.4.8 Class Transaction

26.4.9 Class BalanceInquiry

26.4.10 Class Withdrawal

26.4.11 Class Deposit

26.4.12 Class ATMCaseStudy

26.5 Wrap-Up

27 Java Platform Module System

27.1 Introduction

27.2 Module Declarations

27.2.1 requires

27.2.2 requires transitivelyImplied Readability

27.2.3 exports and exportsto

27.2.4 uses

27.2.5 provideswith

27.2.6 open, opens and opensto

27.2.7 Restricted Keywords

27.3 Modularized Welcome App

27.3.1 Welcome Apps Structure

Table of Contents

27.3.2 Class Welcome

27.3.3 module-info.java

27.3.4 Module-Dependency Graph

27.3.5 Compiling a Module

27.3.6 Running an App from a Modules Exploded Folders

27.3.7 Packaging a Module into a Modular JAR File

27.3.8 Running the Welcome App from a Modular JAR File

27.3.9 Aside: Classpath vs. Module Path

27.4 Creating and Using a Custom Module

27.4.1 Exporting a Package for Use in Other Modules

27.4.2 Using a Class from a Package in Another Module

27.4.3 Compiling and Running the Example

27.4.4 Packaging the App into Modular JAR Files

27.4.5 Strong Encapsulation and Accessibility

27.5 Module-Dependency Graphs: A Deeper Look

27.5.1 java.sql

27.5.2 java.se

27.5.3 Browsing the JDK Module Graph

27.5.4 Error: Module Graph with a Cycle

27.6 Migrating Code to Java 9

27.6.1 Unnamed Module

27.6.2 Automatic Modules

27.6.3 jdeps: Java Dependency Analysis

27.7 Resources in Modules; Using an Automatic Module

27.7.1 Automatic Modules

27.7.2 Requiring Multiple Modules

27.7.3 Opening a Module for Reflection

27.7.4 Module-Dependency Graph

27.7.5 Compiling the Module

Table of Contents

27.7.6 Running a Modularized App

27.8 Creating Custom Runtimes with jlink

27.8.1 Listing the JREs Modules

27.8.2 Custom Runtime Containing Only java.base

27.8.3 Creating a Custom Runtime for the Welcome App

27.8.4 Executing the Welcome App Using a Custom Runtime

27.8.5 Using the Module Resolver on a Custom Runtime

27.9 Services and ServiceLoader

27.9.1 Service-Provider Interface

27.9.2 Loading and Consuming Service Providers

27.9.3 uses Module Directive and Service Consumers

27.9.4 Running the App with No Service Providers

27.9.5 Implementing a Service Provider

27.9.6 provideswith Module Directive and Declaring a Service Provider

27.9.7 Running the App with One Service Provider

27.9.8 Implementing a Second Service Provider

27.9.9 Running the App with Two Service Providers

27.10 Wrap-Up

28 Additional Java 9 Topics

28.1 Introduction

28.2 Recap: Java 9 Features Covered in Earlier Chapters

28.3 New Version String Format

28.4 Regular Expressions: New Matcher Class Methods

28.4.1 Methods appendReplacement and appendTail

28.4.2 Methods replaceFirst and replaceAll

28.4.3 Method results

28.5 New Stream Interface Methods

28.5.1 Stream Methods takeWhile and dropWhile

Table of Contents

28.5.2 Stream Method iterate

28.5.3 Stream Method ofNullable

28.6 Modules in JShell

28.7 JavaFX 9 Skin APIs

28.8 Other GUI and Graphics Enhancements

28.8.1 Multi-Resolution Images

28.8.2 TIFF Image I/O

28.8.3 Platform-Specific Desktop Features

28.9 Security Related Java 9 Topics

28.9.1 Filter Incoming Serialization Data

28.9.2 Create PKCS12 Keystores by Default

28.9.3 Datagram Transport Layer Security (DTLS)

28.9.4 OCSP Stapling for TLS

28.9.5 TLS Application-Layer Protocol Negotiation Extension

28.10 Other Java 9 Topics

28.10.1 Indify String Concatenation

28.10.2 Platform Logging API and Service

28.10.3 Process API Updates

28.10.4 Spin-Wait Hints

28.10.5 UTF-8 Property Resource Bundles

28.10.6 Use CLDR Locale Data by Default

28.10.7 Elide Deprecation Warnings on Import Statements

28.10.8 Multi-Release JAR Files

28.10.9 Unicode 8

28.10.10 Concurrency Enhancements

28.11 Items Removed from the JDK and Java 9

28.12 Items Proposed for Removal from Future Java Versions

28.12.1 Enhanced Deprecation

28.12.2 Items Likely to Be Removed in Future Java Versions

Table of Contents

28.12.3 Finding Deprecated Features

28.12.4 Java Applets

28.13 Wrap-Up

A: Operator Precedence Chart

B: ASCII Character Set

C: Keywords and Reserved Words

D: Primitive Types

E: Bit Manipulation

E.1 Introduction

E.2 Bit Manipulation and the Bitwise Operators

E.3 BitSet Class

F: Labeled break and continue Statements

F.1 Introduction

F.2 Labeled break Statement

F.3 Labeled continue Statement

Index