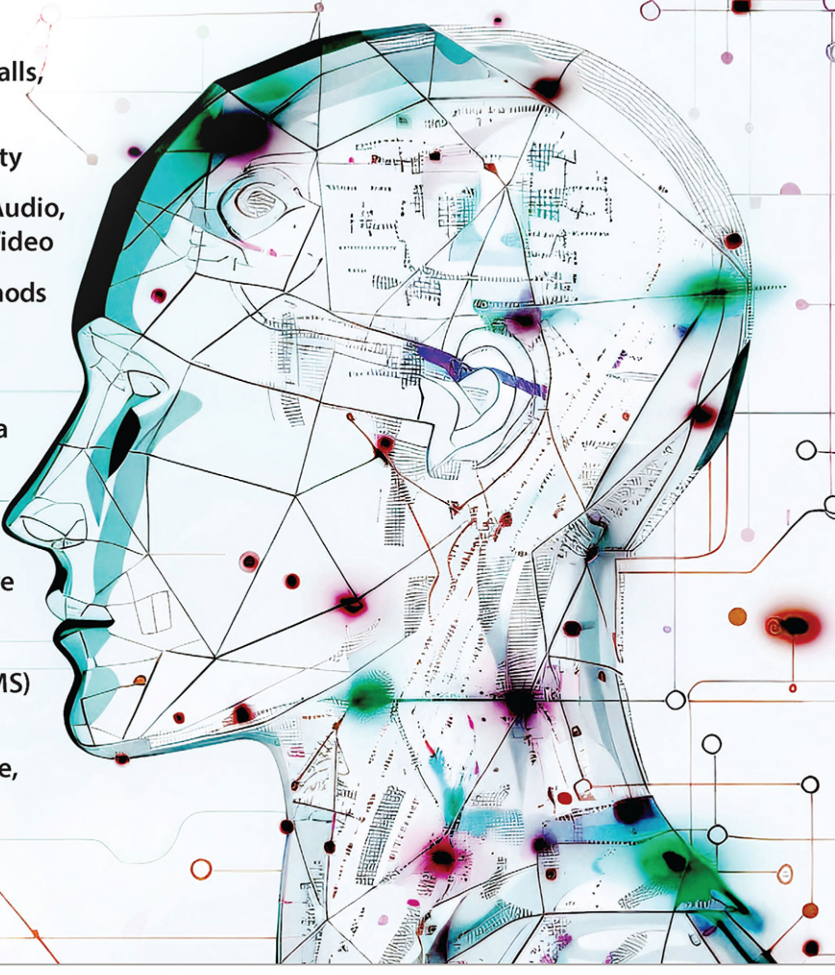
DEITEL® DEVELOPER SERIES

FIFTH EDITION

Java for Programmers *with* Generative AI



- GenAI Prompt Engineering, API Calls, 600 GenAI Exercises
- ChatGPT, Gemini, Claude, Perplexity
- Multimodal: Text, Code, Images, Audio, Speech-to-Text, Text-to-Speech, Video
- Generics: Collections, Classes, Methods
- Functional Programming: Lambdas & Streams
- JavaFX: GUI, Graphics, Multimedia
- Concurrency: Parallel Streams, Virtual Threads, Structured Concurrency, Scoped Values, Concurrent Collections, Multi-Core
- Database: JDBC, SQL, SQLite
- Java Platform Module System (JPMS)
- Objects Natural: Java API, String, BigInteger, BigDecimal, Date/Time, Cryptography, ArrayList, Regex, JSON, CSV, Web Services
- JShell for Python-Like Interactivity

PAUL DEITEL • HARVEY DEITEL

Generative AI Innovations in *Java for Programmers, 5e*

Fully Coded GenAI Case Studies

Chapter 19 presents the following **code examples that interact with OpenAI's APIs**: **Text Summarization**, **Sentiment Analysis**, **Accessible Image Descriptions**, **Language Detection & Translation**, **Java Code Generation**, **Named-Entity Recognition & Structured Outputs**, **Speech-to-Text**, **Text-to-Speech**, **Image Generation**, **Creating Closed Captions for a Video**, **Moderation**.

GenAI Prompt Exercises

We fed the complete list of all the book's **approximately 600 genAI exercises** (a 100+ page PDF) to **ChatGPT**, **Gemini**, **Claude** and **Perplexity**, asking them to categorize the kinds of things we do in those exercises. Next, we fed their categorized lists to the four genAIs, asking them to summarize the summaries, and we chose the best one—**Claude** in this case:

- **Code Generation and Implementation**—Writing new Java programs from specifications. Implementing specific features, algorithms and APIs. Creating test programs and practical applications. Generating solutions for basic and advanced tasks.
- **Code Refactoring and Enhancement**—Modernizing code. Improving code structure, readability, and maintainability. Converting between different approaches while maintaining functionality. Improving performance.
- **Educational Content**—Creating tutorials, exercises, and learning materials. Further exploring complex concepts. Developing programming exercises. Writing comprehensive documentation and guides.
- **Technical Analysis**—Analyzing code behavior and feature implementations. Comparing different approaches, tools, and frameworks. Evaluating trade-offs in design decisions. Breaking down complex technical concepts.
- **Best Practices and Standards**—Implementing coding standards and design patterns. Addressing security considerations. Optimizing performance. Following Java development best practices.
- **Technology Evaluation**—Comparing libraries, tools, and frameworks. Assessing the pros and cons of different approaches. Making informed technology choices. Exploring new features and updates.
- **Debugging and Error Handling**—Finding and fixing syntax and logical errors. Implementing exception handling. Improving fault tolerance. Preventing common pitfalls.
- **API and Library Integration**—Working with Java APIs and external libraries. Understanding API features and capabilities. Implementing integration techniques. Creating API documentation and tutorials.
- **Real-world Applications**—Developing practical use cases and industry applications. Creating interactive applications (GUIs, games, multimedia). Implementing real-world scenarios. Building sample projects.
- **Performance Optimization**—Analyzing and improving performance. Optimizing resource usage. Conducting benchmarks. Implementing efficiency improvements.
- **Creative Development**—Building multimodal applications. Creating visualizations. Generating test scenarios and sample data. Developing unique use cases.

GenAI API-Based Java Programming Exercises

Chapter 19, Building API-Based Java Generative AI Applications, suggests challenging project exercises like creating genAI multimedia apps that can debate one another and using genAIs to build and solve crossword puzzles. We fed the 94 exercises into the genAIs, asking for a categorized summary of them, then summarized the summaries. Here's what **Claude** produced:

- **Multimodal Applications**— Building comprehensive tools that combine text, image, audio, speech and video capabilities. Creating integrated experiences like interactive books. Developing multimedia educational content.
- **Text-Based Applications**—Document processing (indexing, summarization, exploration). Creative writing (stories, poetry, debates). Language tools (translation, tone rewriting). Professional document creation (resumes, presentations). Structured outputs.
- **Image Processing Applications**—Generative art and design (logos, fashion, floor plans). Technical visualization (UML diagrams). Image analysis and recognition.
- **Audio and Music Applications**—Speech processing (transcription, voice cloning). Music generation (MIDI, Magenta AI). Multilingual audio applications. Podcast and audio content analysis.
- **Educational Tools**—Programming tutors (Java, coding exercises). Subject-specific learning aids (math). Course content creation. Interactive educational experiences.
- **Gaming and Puzzle Applications**—Puzzle generators and solvers. Interactive game development.
- **Video**—Investigating and experimenting with generative AI video creation tools.
- **Chatbot Development**—Character-based chat experiences. Specialized domain experts.
- **Research and Analysis Tools**—Medical applications (researching drug discovery and personalized medicine). AI capability exploration. Text detection and analysis. Educational research.
- **Creative Applications**—Children's book creation. Interactive storytelling. Artistic content generation. Creative writing.
- **Practical Tools and Utilities**—Document generators. Translation services. Content summarizers. Professional tools (resume filters, presentation creators).

Java for Programmers: with Generative AI

Table of Contents

Cover

Half Title

Title Page

Copyright Page

Contents

Preface

Before You Begin

1 Intro, Test-Driving a Java Application, and Generative AI

- 1.1 Introduction
- 1.2 Hardware
- 1.3 Java
- 1.4 A Brief Review of Object Orientation
- 1.5 Java Application Programming Interface (API) and Open-Source Libraries
- 1.6 Typical Java Program Development Environment
- 1.7 Test-Driving a Java Program with the Java Development Kit (JDK)
  - 1.7.1 Compiling and Running the Program with javac and java
  - 1.7.2 Running the Programs Source-Code File Directly with the java Command
- 1.8 Internet, World Wide Web, the Cloud and IoT
  - 1.8.1 The Internet: A Network of Networks
  - 1.8.2 The World Wide Web: Making the Internet User-Friendly

Pearson

Table of Contents

1.8.3 The Cloud

1.8.4 The Internet of Things (IoT)

1.8.5 Edge Computing

1.8.6 Mashups

1.9 Metaverse

1.9.1 Virtual Reality (VR)

1.9.2 Augmented Reality (AR)

1.9.3 Mixed Reality

1.9.4 Blockchain

1.9.5 Bitcoin and Cryptocurrency

1.9.6 Ethereum

1.9.7 Non-Fungible Tokens (NFTs)

1.9.8 Web3

1.10 Software Development Technologies

1.11 Data Analytics and Data Science

1.12 How Big Is Big Data?

1.12.1 Big-Data Analytics

1.12.2 Data Science and Big Data Are Making a Difference: Use Cases

1.13 AIat the Intersection of Computer Science and Data Science

1.13.1 Artificial Intelligence (AI)

1.13.2 Artificial General Intelligence (AGI)

1.13.3 Artificial Intelligence Milestones

1.13.4 Machine Learning

1.13.5 Deep Learning

1.13.6 Reinforcement Learning

1.14 Generative AI

1.15 Wrap-Up

2 Intro to Java Programming

Pearson

Table of Contents

2.1 Introduction

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

2.2.1 Compiling the Program

2.2.2 Executing the Program

2.3 Modifying Your First Program

2.4 Displaying Text with printf

2.5 Another Program: Adding Integers

2.6 Arithmetic

2.7 Decision Making: Equality and Relational Operators

2.8 Objects-Natural Case Study: Creating and Using Objects of the Java APIs String Class

2.9 Wrap-Up

3 Control Statements: Part 1

3.1 Introduction

3.2 Control Structures

3.2.1 Sequence Structure

3.2.2 Selection Statements

3.2.3 Iteration Statements

3.2.4 Summary of Control Statements

3.3 if Single-Selection Statement

3.4 ifelse Double-Selection Statement

3.4.1 Nested ifelse Statements

3.4.2 Blocks

3.4.3 Conditional Operator (?:)

3.5 while Iteration Statement

3.6 Counter-Controlled Iteration

3.6.1 Implementing the Algorithm

3.6.2 Integer Division and Truncation

Pearson

Table of Contents

3.6.3 Arithmetic Overflow

3.6.4 Input Validation

3.7 Sentinel-Controlled Iteration

3.7.1 Implementing Sentinel-Controlled Iteration

3.7.2 Explicitly and Implicitly Converting Between Primitive Types

3.7.3 Formatting Floating-Point Numbers

3.8 Nested Control Statements

3.8.1 Problem Statement

3.8.2 Implementing the Program

3.9 Compound Assignment Operators

3.10 Increment and Decrement Operators

3.11 Primitive Types

3.12 Objects-Natural Case Study: Super-Sized Integers

3.13 Wrap-Up

4 Control Statements: Part 2

4.1 Introduction

4.2 Essentials of Counter-Controlled Iteration

4.3 for Iteration Statement

4.4 Examples Using the for Statement

4.5 Summing the Even Integers from 2 to 20 with the for Statement

4.6 Calculating Compound Interest with the for Statement

4.7 dowhile Iteration Statement

4.8 switch Multiple-Selection Statement

4.9 break and continue Statements

4.10 Logical Operators

4.10.1 Conditional AND (&&) Operator

4.10.2 Conditional OR (||) Operator

Table of Contents

4.10.3 Short-Circuit Evaluation of Complex Conditions

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

4.10.5 Boolean Logical Exclusive OR (^) Operator

4.10.6 Logical Complement (!) Operator

4.10.7 Example: Generating Logical-Operator Truth Tables

4.11 Objects-Natural Case Study: Precise Monetary Calculations with Java API Class BigDecimal

4.12 Wrap-Up

5 Methods

5.1 Introduction

5.2 Declaring Methods

5.3 Notes on Declaring and Using Methods

5.4 Case Study: Die Rolling Simulation with Random-Number Generation

5.5 Case Study: A Game of Chance; Introducing enums

5.6 Scope of Declarations

5.7 Method Overloading

5.8 Class Math: static Methods and Variables

5.9 Java API Packages

5.10 Method-Call Stack and Activation Records

5.11 Argument Promotion and Casting

5.12 Objects-Natural Case Study: Java Date/Time API

5.12.1 Data/Time Manipulations

5.12.2 Running the Program for a Different Locale Than Your Own

5.13 Wrap-Up

6 Arrays and ArrayLists

6.1 Introduction

Pearson

Table of Contents

6.2 Primitive Types vs. Reference Types

6.3 Arrays

6.4 Declaring and Creating Arrays

6.5 Creating and Initializing an Array

6.6 Array Initializers

6.7 Calculating Array Element Values

6.8 Totaling Array Elements

6.9 Intro to Visualization: Using a Bar Chart to Display Array Data Graphically

6.10 Using Array Elements as Counters

6.11 Analyzing Survey Results

6.12 Exception Handling

6.12.1 The try Statement

6.12.2 The catch Block

6.13 Enhanced for Statement: Totaling Array Elements

6.14 Passing Arrays to Methods

6.15 Pass-By-Value vs. Pass-By-Reference

6.16 Multidimensional Arrays

6.17 Variable-Length Argument Lists

6.18 Command-Line Arguments

6.19 Class Arrays

6.20 Objects-Natural Case Study: Intro to Collections and Class ArrayList

6.21 Wrap-Up

7 Strings, NLP and Regex: Generative AI Foundations

7.1 Introduction

7.2 Fundamentals of Characters and Strings

Pearson

Table of Contents

7.3 Class String

- 7.3.1 Creating String Objects
- 7.3.2 String Methods length, charAt and getChars
- 7.3.3 Comparing Strings
- 7.3.4 Locating Characters and Substrings in Strings
- 7.3.5 Extracting Substrings from Strings
- 7.3.6 Concatenating Strings
- 7.3.7 Miscellaneous String Methods
- 7.3.8 String Method valueOf

7.4 Class StringBuilder

- 7.4.1 Creating StringBuilder Objects
- 7.4.2 Methods length, capacity, setLength and ensureCapacity
- 7.4.3 Methods charAt, setCharAt, getChars and reverse
- 7.4.4 append Methods
- 7.4.5 Insertion and Deletion Methods

7.5 Class Character

- 7.5.1 Character Testing and Case Conversion Methods
- 7.5.2 Character/Digit Conversions
- 7.5.3 Other Character Methods

7.6 Tokenizing Strings

7.7 Intro to Natural Language Processing (NLP)at the Root of Generative AI

7.8 Objects-Natural Case Study: Intro to Regular Expressions in NLP

- 7.8.1 Matching Complete Strings to Patterns
- 7.8.2 Replacing Substrings
- 7.8.3 Searching for Matches with Classes Pattern and Matcher
- 7.8.4 Simple Data Wrangling Steps Used to Prepare Text for Training NLP and Generative AI Models

7.9 Objects-Natural Security Case Study:

Pearson

Table of Contents

pMa5tfEKwk59dTvC04Ft1IFQz9mEXnkfYXZwxk4ujGE=

7.10 Wrap-Up

8 Real-World Modeling with Custom Classes

8.1 Introduction

8.2 Instance Variables, set Methods and get Methods

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

8.2.2 AccountTest Class That Creates and Uses an Account Object

8.2.3 Compiling and Executing a Program with Multiple Classes

8.2.4 Account UML Class Diagram

8.2.5 Notes on Class AccountTest

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

8.3 Default and Explicit Instance Variable Initialization

8.4 Account Class: Initializing Objects with Constructors

8.4.1 Declaring an Account Constructor for Custom Object Initialization

8.4.2 Class AccountTest: Initializing New Account Objects

8.5 Account Class with a Balance

8.5.1 Account Class with a BigDecimal balance Instance Variable

8.5.2 AccountTest Class

8.6 Case Study: Card Shuffling and Dealing Simulation

8.7 Case Study: Time Class

8.8 Controlling Access to Members

8.9 Referring to the Current Objects Members with the this Reference

8.10 Case Study: Time Class Overloaded Constructors

8.11 Default and No-Argument Constructors

8.12 Notes on Set and Get Methods

8.13 Composition

8.14 enum Types

Table of Contents

8.15 Garbage Collection

8.16 static Class Members

8.17 static Import

8.18 final Instance Variables

8.19 Package Access

8.20 record Classes

8.20.1 Pattern Matching for switch Expressions

8.20.2 Decomposing records in switch Expressions

8.21 Wrap-Up

9 Real-World Modeling with Inheritance, Polymorphism & Interfaces

9.1 Introduction

9.2 Superclasses and Subclasses

9.3 Relationship Between Superclasses and Subclasses

9.3.1 Creating and Using a SalariedEmployee Class

9.3.2 Creating a SalariedEmployee/SalariedCommissionEmployee Inheritance Hierarchy

9.4 Class Object

9.5 Intro to Polymorphism: Polymorphic Video Game

9.6 Demonstrating Polymorphic Behavior

9.7 abstract Classes and Methods

9.8 Case Study: Payroll System Using Polymorphism

9.8.1 abstract Superclass Employee

9.8.2 Concrete Subclass SalariedEmployee

9.8.3 Concrete Subclass CommissionEmployee

9.8.4 Polymorphic Processing, Operator instanceof and Downcasting

9.9 final Methods and Classes

Pearson

Table of Contents

9.10 Issues with Constructors Calling Instance Methods

9.11 Creating and Using Interfaces

- 9.11.1 Developing a Payable Hierarchy
- 9.11.2 Interface Payable
- 9.11.3 Class Invoice
- 9.11.4 Modifying Class Employee to Implement Interface Payable
- 9.11.5 Using Interface Payable to Process Invoices and Employees Polymorphically
- 9.11.6 Some Common Java API Interfaces

9.12 Other Interface Features

- 9.12.1 default Interface Methods
- 9.12.2 static Interface Methods
- 9.12.3 Functional Interfaces
- 9.12.4 private Interface Methods

9.13 Program to an Interface, Not an Implementation

- 9.13.1 CompensationModel Interface
- 9.13.2 An Employee Has a CompensationModelComposition and Dependency Injection
- 9.13.3 CompensationModel Implementations
- 9.13.4 Testing the CompensationModel Hierarchy
- 9.13.5 Dependency Injection Design Benefits
- 9.13.6 interfaces vs. abstract Classes

9.14 sealed Classes and Interfaces

9.15 private Constructors

9.16 protected Members

9.17 Wrap-Up

10 Exception Handling: A Deeper Look

10.1 Introduction

10.2 Example: Divide by Zero without Exception Handling

Table of Contents

10.3 Example: Handling ArithmeticExceptions and InputMismatchExceptions
10.4 Java Exception Hierarchy
10.5 Checked vs. Unchecked Exceptions
10.6 finally Block
10.7 Stack Unwinding and Obtaining Information from an Exception
10.8 Chained Exceptions
10.9 Declaring Custom Exceptions
10.10 Preconditions and Postconditions
10.11 Assertions
10.12 try-with-Resources Statement: Automatic Resource Deallocation
10.13 Unnamed Variables in catch Handlers
10.14 Wrap-Up
11 Files, I/O Streams, JSON Serialization & CSV Files
11.1 Introduction
11.2 Files and Streams
11.3 Using NIO Classes and Interfaces to Get File and Directory Information
11.4 Sequential Text Files
11.4.1 Creating a Sequential Text File
11.4.2 Reading Data from a Sequential Text File
11.4.3 Updating Sequential Files
11.5 Case Study: JavaScript Object Notation (JSON) Serialization
11.5.1 Serializing a List<Account>
11.5.2 Deserializing a List<Account>
11.6 Case Study: Processing a JSON Response from a Web Service

Table of Contents

11.7 Case Study: Creating and Reading CSV Files

11.7.1 Creating a CSV File

11.7.2 Reading a CSV File

11.8 Case Study: Reading and Analyzing a CSV File Containing the Titanic Disaster Dataset

11.9 Objects-Natural Security Case Study: RSA Public-Key Cryptography

11.10 Wrap-Up

12 Generic Collections

12.1 Introduction

12.2 Collections Overview

12.3 Type-Wrapper Classes

12.4 Boxing and Unboxing

12.5 Lists

12.5.1 ArrayList and Iterator

12.5.2 LinkedList

12.5.3 Views into Collections and Arrays Method asList

12.6 Collections Methods

12.6.1 Method sort

12.6.2 Method shuffle

12.6.3 Methods reverse, fill, copy, max and min

12.6.4 Method binarySearch

12.6.5 Methods addAll, frequency and disjoint

12.7 Class PriorityQueue and Interface Queue

12.8 Hash Tables

12.9 Sets

12.10 Maps

12.11 Convenience Factory Methods for Creating Immutable

Table of Contents

Collections

12.12 Concurrent Collections

12.13 Wrap-Up

13 Generic Classes and Methods: A Deeper Look

13.1 Introduction

13.2 Motivation for Generic Methods

13.3 Generic Methods: Implementation and Compile-Time Translation

13.3.1 Generic Method printArray

13.3.2 Type Parameter Section of a Generic Method

13.3.3 Testing the Generic Method

13.3.4 Erasure at Compilation Time

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

13.5 Overloading Generic Methods

13.6 Generic Classes

13.6.1 Parameterized Types

13.6.2 Implementing a Generic Stack Class

13.6.3 Testing the Generic Stack Class

13.6.4 Refactoring the Test Program to Manipulate Stack<E> Objects with Generic Methods

13.7 Wildcards in Methods That Accept Type Parameters

13.7.1 Summing a List<Number>

13.7.2 Reimplementing Method sum with a Wildcard Type Argument in Its Parameter

13.8 Wrap-Up

14 Functional Programming with Lambdas & Streams

14.1 Introduction

Pearson

Table of Contents

14.2 Streams and Reduction

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

14.2.2 External Iteration with for Is Error Prone

14.2.3 Summing with a Stream and Reduction

14.2.4 Internal Iteration

14.3 Mapping and Lambdas

14.3.1 Lambda Expressions

14.3.2 Lambda Syntax

14.3.3 Intermediate and Terminal Operations

14.4 Filtering

14.5 How Elements Move Through Stream Pipelines

14.6 Method References

14.6.1 Creating an IntStream of Random Values

14.6.2 Performing a Task on Each Stream Element with forEach and a Method Reference

14.6.3 Mapping Integers to String Objects with mapToObj

14.6.4 Concatenating Strings with collect

14.7 IntStream Operations

14.7.1 Creating an IntStream and Displaying Its Values

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

14.7.3 Terminal Operation reduce

14.7.4 Sorting IntStream Values

14.8 Functional Interfaces

14.9 Lambdas: A Deeper Look

14.10 Stream<Integer> Manipulations

14.10.1 Creating a Stream<Integer>, Sorting It and Collecting the Results

14.10.2 Filtering a Stream and Storing the Results for Later Use

14.10.3 Filtering and Sorting a Stream and Collecting the Results

14.10.4 Sorting Previously Collected Results

Pearson

Table of Contents

14.11 Stream<String> Manipulations
14.11.1 Mapping Strings to Uppercase
14.11.2 Filtering Strings Then Sorting Them in Case-Insensitive Ascending Order
14.11.3 Filtering Strings Then Sorting Them in Case-Insensitive Descending Order
14.12 Stream<Employee> Manipulations
14.12.1 Creating and Displaying a List<Employee>
14.12.2 Filtering Employees with Salaries in a Specified Range
14.12.3 Sorting Employees By Multiple Fields
14.12.4 Mapping Employees to Unique-Last-Name Strings
14.12.5 Grouping Employees By Department
14.12.6 Counting the Number of Employees in Each Department
14.12.7 Summing and Averaging Employee Salaries
14.13 Creating a Stream<String> from a File
14.14 Streams of Random Values
14.15 Infinite Streams
14.16 Additional Notes on Interfaces
14.17 Wrap-Up
15 JavaFX Graphical User Interfaces: Part 1
15.1 Introduction
15.2 JavaFX Scene Builder
15.3 JavaFX Application Window Structure
15.4 Welcome Application: Displaying Text and an Image
15.4.1 Opening Scene Builder and Creating the File Welcome.fxml
15.4.2 Adding an Image to the Folder Containing Welcome.fxml
15.4.3 Creating a VBox Layout Container
15.4.4 Configuring the VBox Layout Container

Table of Contents

15.4.5 Adding and Configuring a Label

15.4.6 Adding and Configuring an ImageView

15.4.7 Previewing the Welcome GUI

15.5 Tip Calculator Application: Intro to Event Handling

15.5.1 Test-Driving the Tip Calculator

15.5.2 Technologies Overview

15.5.3 Building the Applications GUI

15.5.4 TipCalculator Class

15.5.5 TipCalculatorController Class; Implementing an Event Handler with a Lambda

15.6 Features Covered in the Other JavaFX Chapters

15.7 Wrap-Up

16 JavaFX GUI: Part 2

16.1 Introduction

16.2 Laying Out Nodes in a Scene Graph

16.3 Painter Application: RadioButtons, Mouse Events and Shapes

16.3.1 Technologies Overview

16.3.2 Creating the Painter.fxml File

16.3.3 Building the GUI

16.3.4 Painter Subclass of Application

16.3.5 PainterController Class

16.4 Color Chooser Application: Property Bindings and Property Listeners

16.4.1 Technologies Overview

16.4.2 Building the GUI

16.4.3 ColorChooser Subclass of Application

16.4.4 ColorChooserController Class

16.5 Cover Viewer Application: Data-Driven GUIs with JavaFX Collections

Table of Contents

16.5.1 Technologies Overview

16.5.2 Building the GUI

16.5.3 CoverViewer Subclass of Application

16.5.4 CoverViewerController Class

16.6 Cover Viewer Application: Customizing ListView Cells

16.6.1 Technologies Overview

16.6.2 Copying the CoverViewer Application

16.6.3 ImageTextCell Custom ListView Cell Layout

16.6.4 CoverViewerController Class

16.7 FileChooser and DirectoryChooser Dialogs

16.8 Other JavaFX Capabilities and JavaFX Accessibility

16.9 JavaFX Updates

16.10 JavaFX Resources and Libraries

16.11 Wrap-Up

17 JavaFX Graphics and Multimedia

17.1 Introduction

17.2 Controlling Fonts with Cascading Style Sheets (CSS)

17.2.1 CSS That Styles the GUI

17.2.2 FXML That Defines the GUIIntroduction to XML Markup

17.2.3 Referencing the CSS File from FXML

17.2.4 Specifying the VBoxs Style Class

17.2.5 Programmatically Loading CSS

17.3 Displaying Two-Dimensional Shapes

17.3.1 Defining Two-Dimensional Shapes with FXML

17.3.2 CSS That Styles the Two-Dimensional Shapes

17.4 Polylines, Polygons and Paths

17.4.1 GUI and CSS

17.4.2 PolyShapesController Class

Pearson

Table of Contents

17.5 Transforms

17.6 Playing Video with Media, MediaPlayer and MediaView

17.6.1 VideoPlayer GUI

17.6.2 VideoPlayerController Class

17.7 Transition Animations

17.7.1 TransitionAnimations.fxml

17.7.2 TransitionAnimationsController Class

17.8 Timeline Animations

17.9 Frame-by-Frame Animation with AnimationTimer

17.10 CSS Transitions

17.11 Drawing on a Canvas

17.12 Three-Dimensional Shapes

17.13 FXGL: A Brief Intro to Game Programming with JavaFX

17.14 Wrap-Up

18 Concurrency: Platform Threads to Virtual Threads

18.1 Introduction

18.2 sort/parallelSort Timings with the Date/Time API

18.3 Sequential vs. Parallel Streams

18.4 Creating and Executing Platform Threads with the Executor Framework

18.5 Project Loom Overview

18.6 Creating and Executing Virtual Threads with the Executor Framework

18.6.1 Launching Virtual Threads with a Thread-Per-Task Executor

18.6.2 Other Ways to Launch Virtual Threads

18.7 Profiling Platform vs. Virtual Threads

18.7.1 Future and Callable Interfaces

Pearson

Table of Contents

18.7.2 Threading Approaches Well Use

18.7.3 Launching and Timing Large Numbers of Concurrent Tasks

18.7.4 Testing the Threading Approaches

18.7.5 Performance Analysis

18.8 Structured Concurrency and Scoped Values

18.9 Thread Synchronization Overview

18.10 Producer/Consumer Relationship with ArrayBlockingQueue

18.10.1 Synchronization and State Dependence

18.10.2 Class ArrayBlockingQueue

18.10.3 Overview of the Producer/Consumer Example

18.10.4 Interface Buffer

18.10.5 Class BlockingBuffer

18.10.6 Class BlockingBuffer Test

18.10.7 Bounded Buffers

18.10.8 Other Concurrent Collections

18.11 Multithreading in JavaFX

18.11.1 Performing Computations in a Worker Thread: Fibonacci Numbers

18.11.2 Processing Intermediate Results: Sieve of Eratosthenes

18.12 Wrap-Up

19 Building API-Based Java Generative AI Applications

19.1 Introduction

19.2 OpenAI APIs

19.2.1 Some of OpenAIs Models

19.2.2 OpenAI API Capabilities

19.2.3 Get an OpenAI Developer Account

19.2.4 OpenAI Developer API Key

19.3 Setting Up a Java Environment

19.3.1 Simple-OpenAI Open Source Library: Connecting to | OpenAI from

Pearson

Table of Contents

Java
19.3.2 IntelliJ Maven-Based Java Project
19.4 Text Generation Via Chat Completions
19.4.1 Text Summarization
19.4.2 Sentiment Analysis
19.4.3 Accessible Image Descriptions
19.4.4 Language Detection and Translation
19.4.5 Code Generation
19.4.6 Named Entity Recognition (NER) and Structured Outputs
19.5 Speech Synthesis and Speech Recognition
19.5.1 English Speech-to-Text for Audio Transcription
19.5.2 Text-to-Speech
19.6 Image Generation
19.7 Video
19.7.1 Generating Video Closed Captions
19.7.2 Sora
19.7.3 Other Video GenAIs
19.8 Moderation
19.9 Class OpenAIUtilities
19.9.1 Builder Design Pattern
19.9.2 package and import Statements
19.9.3 Nested record Class Message
19.9.4 SimpleOpenAI Client Object
19.9.5 chat Method
19.9.6 Helper Method makeChatMessages
19.9.7 describeImage Method
19.9.8 translate Method
19.9.9 chatWithStructuredOutput Method
19.9.10 speechToText Method

Table of Contents

19.9.11 textToSpeech Method
19.9.12 image Method
19.9.13 speechToVTT Method
19.9.14 checkPrompt Method
19.10 Wrap-Up
20 Accessing Databases with JDBC and SQLite
20.1 Introduction
20.2 Relational Databases
20.3 Setting Up the SQLite RDBMS
20.4 A books Database
20.4.1 SELECT Queries
20.4.2 WHERE Clause
20.4.3 ORDER BY Clause
20.4.4 Merging Data from Multiple Tables with INNER JOIN
20.4.5 INSERT INTO Statement
20.4.6 UPDATE Statement
20.4.7 DELETE FROM Statement
20.5 Connecting to and Querying a Database with JDBC
20.5.1 Automatic Driver Discovery
20.5.2 Connecting to the Database
20.5.3 Creating a Statement for Executing Queries
20.5.4 Executing a Query
20.5.5 Processing a Querys ResultSet
20.6 Querying the books Database
20.6.1 DisplayQueryResults Apps GUI
20.6.2 DisplayQueryResultsController Class
20.7 PreparedStatements
20.7.1 AddressBook App That Uses PreparedStatements

Table of Contents

20.7.2 record Class Person

20.7.3 Class PersonQueries

20.7.4 AddressBook GUI

20.7.5 Class AddressBookController

20.8 Stored Procedures

20.9 Transaction Processing

20.10 Wrap-Up

21 Java Platform Module System

21.1 Introduction

21.2 Module Declarations

21.2.1 requires

21.2.2 requires transitiveImplied Readability

21.2.3 exports and exportsto

21.2.4 uses

21.2.5 provideswith

21.2.6 open, opens and opensto

21.2.7 Restricted Keywords

21.3 Modularized Welcome App

21.3.1 Welcome Apps Structure

21.3.2 Class Welcome

21.3.3 module-info.java

21.3.4 Module-Dependency Graph

21.3.5 Compiling a Module

21.3.6 Running an App from a Modules Exploded Folders

21.3.7 Packaging a Module into a Modular JAR File

21.3.8 Running the Welcome App from a Modular JAR File

21.3.9 Aside: Classpath vs. Module Path

21.4 Creating and Using a Custom Module

Table of Contents

- 21.4.1 Exporting a Package for Use in Other Modules
- 21.4.2 Using a Class from a Package in Another Module
- 21.4.3 Compiling and Running the Example
- 21.4.4 Packaging the App into Modular JAR Files
- 21.4.5 Strong Encapsulation and Accessibility

21.5 Module-Dependency Graphs: A Deeper Look

- 21.5.1 java.sql
- 21.5.2 java.se
- 21.5.3 Error: Module Graph with a Cycle

21.6 Migrating Code to Modules

- 21.6.1 Unnamed Module
- 21.6.2 Automatic Modules
- 21.6.3 jdeps Tool for Java Dependency Analysis

21.7 Resources in Modules

- 21.7.1 Requiring Multiple Modules
- 21.7.2 Opening a Module for Reflection
- 21.7.3 Compiling the Module
- 21.7.4 Running a Modularized App
- 21.7.5 Module-Dependency Graph

21.8 Creating Custom Runtimes with jlink

- 21.8.1 Custom Runtime Containing Only java.base
- 21.8.2 Creating a Custom Runtime for the Welcome App
- 21.8.3 Executing the Welcome App Using a Custom Runtime
- 21.8.4 Using the Module Resolver on a Custom Runtime

21.9 Services and ServiceLoader

- 21.9.1 Service-Provider Interface
- 21.9.2 Loading and Consuming Service Providers
- 21.9.3 uses Module Directive and Service Consumers
- 21.9.4 Running the App with No Service Providers

Pearson

Table of Contents

21.9.5 Implementing a Service Provider

21.9.6 provideswith Module Directive and Declaring a Service Provider

21.9.7 Running the App with One Service Provider

21.9.8 Implementing a Second Service Provider

21.9.9 Running the App with Two Service Providers

21.10 Wrap-Up

22 Recursion and Big O

22.1 Introduction

22.2 Recursion Concepts

22.3 Recursion Example: Factorials

22.4 Recursion Example: Fibonacci Series

22.5 Recursion vs. Iteration

22.6 Towers of Hanoi

22.7 Fractals

22.7.1 Koch Curve Fractal

22.7.2 (Optional) Case Study: Lo Feather Fractal

22.7.3 (Optional) Fractal App GUI

22.7.4 (Optional) FractalController Class

22.8 Recursive Backtracking

22.9 Big O Notation

22.9.1 O(1) Algorithms

22.9.2 O(n) Algorithms

22.9.3 O(n) Algorithms

22.10 Common Big O Notations

22.11 Wrap-Up

A Introduction to JShell for Interactive Java

A.1 Introduction

Pearson

Table of Contents

A.2 Introduction to JShell

- A.2.1 Starting a JShell Session
- A.2.2 Executing Statements
- A.2.3 Declaring Variables Explicitly
- A.2.4 Listing and Executing Prior Snippets
- A.2.5 Evaluating Expressions and Declaring Variables Implicitly
- A.2.6 Using Implicitly Declared Variables
- A.2.7 Viewing a Variables Value
- A.2.8 Resetting a JShell Session
- A.2.9 Writing Multiline Statements
- A.2.10 Editing Code Snippets
- A.2.11 Exiting JShell

A.3 Command-Line Input in JShell

A.4 Declaring and Using Classes

- A.4.1 Creating a Class in JShell
- A.4.2 Explicitly Declaring Reference-Type Variables
- A.4.3 Creating Objects
- A.4.4 Manipulating Objects
- A.4.5 Creating a Meaningful Variable Name for an Expression
- A.4.6 Saving and Opening Code-Snippet Files

A.5 Discovery with JShell Auto-Completion

- A.5.1 Auto-Completing Identifiers
- A.5.2 Auto-Completing JShell Commands

A.6 Exploring a Classs Members and Viewing Documentation

- A.6.1 Listing Class Maths static Members
- A.6.2 Viewing a Methods Parameters
- A.6.3 Viewing a Methods Documentation
- A.6.4 Viewing a public Fields Documentation
- A.6.5 Viewing a Classs Documentation

Pearson

Table of Contents

A.6.6 Viewing Method Overloads

A.6.7 Exploring Members of a Specific Object

A.7 Declaring Methods

A.7.1 Forward Referencing an Undeclared Method Declaring Method displayCubes

A.7.2 Declaring a Previously Undeclared Method

A.7.3 Testing cube and Replacing Its Declaration

A.7.4 Testing Updated Method cube and Method displayCubes

A.8 Exceptions

A.9 Importing Classes and Adding Packages to the CLASSPATH

A.10 Using an External Editor

A.11 Summary of JShell Commands

A.11.1 Getting Help in JShell

A.11.2 /edit Command: Additional Features

A.11.3 /reload Command

A.11.4 /drop Command

A.11.5 Feedback Modes

A.11.6 Other JShell Features Configurable with /set

A.12 Keyboard Shortcuts for Snippet Editing

A.13 How JShell Reinterprets Java for Interactive Use

A.14 IDE JShell Support

A.15 Wrap-Up

B Formatted Output

B.1 Introduction

B.2 Output with printf

B.3 Integer Formatting

B.4 Floating-Point Number Formatting

B.5 String and Character Formatting

Table of Contents

B.6 Other Conversion Characters

B.7 Field Widths and Precisions

B.8 Formatting Flags

B.9 Argument Indices for Explicit Positioning in Format Strings

B.10 Escape Sequences

B.11 Formatting Strings in Memory

B.12 Wrap-Up

C Number Systems

C.1 Introduction

C.2 Abbreviating Binary Numbers as Octal and Hexadecimal Numbers

C.3 Converting Octal and Hexadecimal Numbers to Binary Numbers

C.4 Converting from Binary, Octal or Hexadecimal to Decimal

C.5 Converting from Decimal to Binary, Octal or Hexadecimal

C.6 Negative Binary Numbers: Twos Complement Notation

Index

Pearson