

ORACLE®
University

Integrated Cloud Applications & Platform Services

Java SE: Programming II

Student Guide - Volume I

D102474GC10

Edition 1.0 | May 2019 | D105722

Learn more from Oracle University at education.oracle.com

ORACLE®

Authors

Kenny Somerville
Anjana Shenoy
Nick Ristuccia

Technical Contributors and Reviewers

Joe Greenwald
Jeffrey Picchione
Joe Boulenouar
Steve Watts
Pete Iaseau
Henry Jen
Nick Ristuccia
Alex Buckley
Vasily Strelnikov
Aurelio García-Ribeyro
Stuart Marks
Geertjan Wielenga
Mike Williams

Editors

Moushmi Mukherjee
Raj Kumar

Graphic Designers

Anne Elizabeth
Yogita Chawdhary
Kavya Bellur

Publishers

Asief Baig
Jayanthi Keshavamurthy

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Contents

1 Introduction

- Course Objectives 1-2
- Introductions 1-4
- Audience 1-5
- Prerequisites 1-6
- Course Roadmap 1-7
- Lesson Format 1-12
- Practice Environment 1-13
- How Do You Learn More After the Course? 1-14
- Additional Resources 1-15
- Summary 1-17
- Practice 1: Overview 1-18

2 Java OOP Review

- Objectives 2-2
- Java Language Review 2-3
- A Simple Java Class: Employee 2-4
- Encapsulation: Private Data, Public Methods 2-5
- Subclassing 2-6
- Constructors in Subclasses 2-7
- Using super 2-8
- Using Access Control 2-9
- Protected Access Control: Example 2-10
- Inheritance: Accessibility of Overriding Methods 2-11
- Final Methods 2-12
- Final Classes 2-13
- Applying Polymorphism 2-14
- Overriding methods of Object Class 2-16
- Overriding methods of Object Class: toString Method 2-17
- Overriding methods of Object Class: equals Method 2-18
- Overriding methods of Object Class: hashCode Method 2-20
- Casting Object References 2-21
- Upward Casting Rules 2-22
- Downward Casting Rules 2-23
- Methods Using Variable Arguments 2-24

Static Imports 2-26
Nested Classes 2-27
Example: Member Class 2-28
What Are Enums? 2-29
Complex Enums 2-30
Methods in Enums 2-31
Summary 2-33
Practice 2: Overview 2-34
Quiz 2-35

3 Exception Handling and Assertions

Objectives 3-2
Error Handling 3-3
Exception Types 3-4
Exception Handling Techniques in Java 3-6
Exception Handling Techniques: try Block 3-7
Exception Handling Techniques: finally Clause 3-8
Exception Handling Techniques: try-with-resources 3-9
Exception Handling Techniques: try-with-resources Improvements 3-10
Exception Handling Techniques: | operator in a catch block 3-12
Exception Handling Techniques: Declaring Exceptions 3-13
Creating Custom Exceptions 3-14
Assertions 3-15
Assertion Syntax 3-16
Internal Invariants 3-17
Control Flow Invariants 3-18
Class Invariants 3-19
Controlling Runtime Evaluation of Assertions 3-20
Summary 3-21
Practice 3: Overview 3-22
Quiz 3-23

4 Java Interfaces

Objectives 4-2
Java Interfaces 4-3
Java SE 7 Interfaces 4-4
Implementing Java SE 7 Interface Methods 4-5
Example: Implementing abstract Methods 4-6
Example: Duplicating Logic 4-7
Implementing Methods in Interfaces 4-8
Example: Implementing default Methods 4-9

Example: Inheriting default Methods 4-10
Example: Overriding a default Method 4-11
What About the Problems of Multiple Inheritance? 4-12
Inheritance Rules of default Methods 4-13
Interfaces Don't Replace Abstract Classes 4-16
What If default Methods Duplicate Logic? 4-17
Duplication Between default Methods 4-18
The Problem with This Approach 4-19
Introducing private Methods in Interfaces 4-20
Example: Using private Methods to Reduce Duplication Between
default Methods 4-21
Types of Methods in Java SE 9 Interfaces 4-22
Anonymous Inner Classes 4-23
Anonymous Inner Class: Example 4-24
Summary 4-25
Practice 4: Overview 4-26
Quiz 4-27

5 Collections and Generics

Objectives 5-2
Type-Wrapper Classes 5-3
Autoboxing and Auto-Unboxing 5-4
Generic Methods 5-5
A Generic Method 5-6
Generic Classes 5-7
Generic Cache Class 5-8
Testing the Generic Cache Class 5-9
Generics with Type Inference Diamond Notation 5-10
Java SE 9: Diamond Notation with Anonymous Inner Classes 5-11
Collections 5-12
Collections Framework in Java 5-13
Benefits of the Collections Framework 5-14
Collection Types 5-15
Key Collections Interfaces 5-16
ArrayList 5-17
ArrayList Without Generics 5-18
Generic ArrayList 5-19
TreeSet: Implementation of Set 5-20
Map Interface 5-21
TreeMap: Implementation of Map 5-22
Stack with Deque: Example 5-23

Ordering Collections 5-24
Comparable: Example 5-25
Comparable Test: Example 5-26
Comparator Interface 5-27
Comparator: Example 5-28
Comparator Test: Example 5-29
Wildcards 5-30
Wildcards: Upper Bound 5-31
Why Use Generics? 5-32
Java SE 9: Convenience Methods for Collections 5-33
of Convenience Method 5-34
Overloading of Method 5-35
ofEntries Method for Maps 5-36
Features of Convenience Methods 5-37
Summary 5-38
Practice 5: Overview 5-39
Quiz 5-40

6 Functional Interfaces and Lambda Expressions

Objectives 6-2
Problem Statement 6-3
RoboCall Class 6-4
RoboCall Every Person 6-5
RoboCall Use Case: Eligible Drivers 6-6
RoboCall Use Case: Eligible Voters 6-7
RoboCall Use Case: Legal Drinking Age 6-8
Solution: Parameterization of Values 6-9
Solution: Parameterized Methods 6-10
Parameters for Age Range 6-11
Using Parameters for Age Range 6-12
Corrected Use Case 6-13
Parameterized Computation 6-14
How To Pass a Function in Java? 6-15
Prior to Java SE 8: Pass a Function Wrapped in an Object 6-16
Prior to SE 8: Abstract Behavior With an Interface 6-17
Prior to SE 8: Replace Implementation Class with Anonymous Inner Class 6-18
Lambda Solution: Replace Anonymous Inner Class with Lambda Expression 6-19
Rewriting the Use Cases Using Lambda 6-20
What Is a Lambda? 6-21
What is a Functional Interface 6-22
Which of These Interfaces Are Functional Interfaces? 6-23

Lambda Expression	6-24
Using Lambdas	6-25
Which of The Following Are Valid Lambda Expressions?	6-26
Lambda Expression: Type Inference	6-27
To Create a Lambda Expression	6-28
Examples of Lambdas	6-29
Statement Lambdas: Lambda with Body	6-30
Examples: Lambda Expression	6-31
Lambda Parameters	6-32
Local-Variable Syntax for Lambda Parameters	6-33
Functional Interfaces: Predicate	6-35
Using Functional Interfaces	6-36
Quiz	6-37
Summary	6-39
Practice 6: Overview	6-40

7 Collections, Streams, and Filters

Objectives	7-2
Collections, Streams, and Filters	7-3
The RoboCall App	7-4
Collection Iteration and Lambdas	7-5
RoboCallTest07: Stream and Filter	7-6
Stream and Filter	7-7
RobocallTest08: Stream and Filter Again	7-8
SalesTxn Class	7-9
Java Streams	7-10
The Filter Method	7-11
Filter and SalesTxn Method Call	7-12
Method References	7-13
Method Chaining	7-14
Pipeline Defined	7-16
Summary	7-17
Practice 7: Overview	7-18

8 Lambda Built-in Functional Interfaces

Objectives	8-2
Built-in Functional Interfaces	8-3
The java.util.function Package	8-4
Example Assumptions	8-5
Predicate	8-6
Predicate: Example	8-7

Consumer 8-8
Consumer: Example 8-9
Function 8-10
Function: Example 8-11
Supplier 8-12
Supplier: Example 8-13
Primitive Interface 8-14
Return a Primitive Type 8-15
Return a Primitive Type: Example 8-16
Process a Primitive Type 8-17
Process Primitive Type: Example 8-18
Binary Types 8-19
Binary Type: Example 8-20
Unary Operator 8-21
UnaryOperator: Example 8-22
Wildcard Generics Review 8-23
A Closer Look at Consumer 8-24
Interface List<E> use of forEach method 8-25
Example of Range of Valid Parameters 8-26
Plant Class Example 8-27
TropicalFruit Class Example 8-29
Use of Generic Expressions and Wildcards 8-30
Consumer andThen method 8-31
Using Consumer.andThen method 8-32
Summary 8-34
Practice 8: Overview 8-35

9 Lambda Operations

Objectives 9-2
Streams API 9-3
Types of Operations 9-4
Extracting Data with Map 9-5
Taking a Peek 9-6
Search Methods: Overview 9-7
Search Methods 9-8
Optional Class 9-9
Short-Circuiting Example 9-10
Stream Data Methods 9-11
Performing Calculations 9-12
Sorting 9-13
Comparator Updates 9-14

Saving Data from a Stream 9-15
Collectors Class 9-16
Quick Streams with Stream.of 9-17
Flatten Data with flatMap 9-18
flatMap in Action 9-19
Summary 9-20
Practice 9: Overview 9-21

10 The Module System

Objectives 10-2
Module System 10-3
Module System: Advantages 10-4
Java Modular Applications 10-5
What Is a Module? 10-6
A Modular Java Application 10-7
Issues with Access Across Nonmodular JARs 10-8
Dependencies Across Modules 10-9
What Is a Module? 10-11
Module Dependencies with requires 10-12
Module Package Availability with exports 10-13
Module Graph 1 10-14
Module Graph 2 10-15
Transitive Dependencies 10-16
Access to Types via Reflection 10-17
Example Hello World Modular Application Code 10-18
Example Hello World Modular File Structure 10-19
Compiling a Modular Application 10-20
Single Module Compilation Example 10-21
Multi Module Compilation Example 10-22
Creating a Modular JAR 10-23
Running a Modular Application 10-24
The Modular JDK 10-25
Java SE Modules 10-26
The Base Module 10-28
Finding the Right Platform Module 10-29
Illegal Access to JDK Internals in JDK 9 10-30
What Is a Custom Runtime Image? 10-31
Link Time 10-32
Using jlink to Create a Runtime Image 10-33
Example: Using jlink to Create a Runtime Image 10-34
Examining the Generated Image 10-35

Modules Resolved in a Custom Runtime Image	10-36
Advantages of a Custom Runtime Image	10-37
JIMAGE Format	10-38
Running the Application	10-39
Summary	10-41
Practice 10: Overview	10-42

11 Migrating to a Modular Application

Objectives	11-2
Topics	11-3
The League Application	11-4
Run the Application	11-5
The Unnamed Module	11-6
Topics	11-7
Top-down Migration and Automatic Modules	11-8
Automatic Module	11-9
Top-Down Migration	11-10
Creating module-info.java—Determining Dependencies	11-11
Check Dependencies	11-12
Library JAR to Automatic Module	11-13
Typical Application Modularized	11-14
Topics	11-15
Bottom-up Migration	11-16
Bottom-Up Migration	11-17
Modularized Library	11-18
Run Bottom-Up Migrated Application	11-19
Fully Modularized Application	11-20
Module Resolution	11-21
Topics	11-22
More About Libraries	11-23
Run Application with Jackson Libraries	11-24
Open Soccer to Reflection from Jackson Libraries	11-25
Topics	11-26
Split Packages	11-27
Splitting a Java 8 Application into Modules	11-28
Java SE 8 Application Poorly Designed with Split Packages	11-29
Migration of Split Package JARs to Java SE 9	11-30
Addressing Split Packages	11-31
Topics	11-32
Cyclic Dependencies	11-33
Addressing Cyclic Dependency 1	11-34

Addressing Cyclic Dependency 2 11-35
Top-down or Bottom-up Migration Summary 11-36
Summary 11-37
Practice 11: Overview 11-38

12 Services in a Modular Application

Objectives 12-2
Topics 12-3
Modules and Services 12-4
Components of a Service 12-5
Produce and Consume Services 12-6
Module Dependencies Without Services 12-7
Service Relationships 12-8
Expressing Service Relationships 12-9
Topics 12-10
Using the Service Type in competition 12-11
Choosing a Provider Class 12-12
Module Dependencies and Services 1 12-14
Module Dependencies and Services 2 12-15
Module Dependencies and Services 3 12-16
Designing a Service Type 12-17
Topics 12-18
TeamGameManager Application with Additional Services 12-19
module-info.java for competition module 12-20
module-info.java for league and knockout modules 12-21
module-info.java for soccer and basketball modules 12-22
Summary 12-23
Practice 12: Overview 12-24

13 Concurrency

Objectives 13-2
Task Scheduling 13-3
Legacy Thread and Runnable 13-4
Extending Thread 13-5
Implementing Runnable 13-6
The java.util.concurrent Package 13-7
Recommended Threading Classes 13-8
java.util.concurrent.ExecutorService 13-9
Example ExecutorService 13-10
Shutting Down an ExecutorService 13-11
java.util.concurrent.Callable 13-12

Example Callable Task	13-13
java.util.concurrent.Future	13-14
Example	13-15
Threading Concerns	13-16
Shared Data	13-17
Problems with Shared Data	13-18
Nonshared Data	13-19
Atomic Operations	13-20
Out-of-Order Execution	13-21
The synchronized Keyword	13-22
synchronized Methods	13-23
synchronized Blocks	13-24
Object Monitor Locking	13-25
Threading Performance	13-26
Performance Issue: Examples	13-27
java.util.concurrent Classes and Packages	13-28
The java.util.concurrent.atomic Package	13-29
java.util.concurrent.CyclicBarrier	13-30
Thread-Safe Collections	13-32
CopyOnWriteArrayList: Example	13-33
Summary	13-34
Practice 13: Overview	13-35
Quiz	13-36

14 Parallel Streams

Objectives	14-2
Streams Review	14-3
Old Style Collection Processing	14-4
New Style Collection Processing	14-5
Stream Pipeline: Another Look	14-6
Styles Compared	14-7
Parallel Stream	14-8
Using Parallel Streams: Collection	14-9
Using Parallel Streams: From a Stream	14-10
Pipelines Fine Print	14-11
Embrace Statelessness	14-12
Avoid Statefulness	14-13
Streams Are Deterministic for Most Part	14-14
Some Are Not Deterministic	14-15
Reduction	14-16
Reduction Fine Print	14-17

Reduction: Example 14-18
A Look Under the Hood 14-24
Illustrating Parallel Execution 14-25
Performance 14-36
A Simple Performance Model 14-37
Summary 14-38
Practice 14: Overview 14-39

15 Terminal Operations: Collectors

Objectives 15-2
Agenda 15-3
Streams and Collectors Versus Imperative Code 15-4
Collection 15-5
Predefined Collectors 15-6
A Simple Collector 15-7
A More Complex Collector 15-8
Agenda 15-9
The Three Argument collect Method of Stream 15-10
The collect Method Used with a Sequential Stream 15-11
The collect Method Used with a Parallel Stream 15-12
The collect Method: Collect to an ArrayList Example 15-13
Agenda 15-14
The Single Argument collect Method of Stream 15-15
Using Predefined Collectors From the Collectors Class 15-16
List of Predefined Collectors 15-17
Stand-Alone Collectors 15-18
Stand-Alone Collector: List all Elements 15-19
maxBy() Example 15-20
Adapting Collector: filtering() 15-21
Composing Collectors : groupingBy() 15-22
Composing Collectors: Using Mapping 15-23
toMap() and Duplicate Keys 15-24
Agenda 15-25
groupingBy and partitioningBy Collectors 15-26
Stand-Alone groupingBy: Person Elements By City 15-27
Stream Operations Or Equivalent Collectors? 15-28
Stream Operations Or Equivalent Collectors with groupingBy 15-29
Stream.count(), Collectors.counting and groupingBy 15-30
Composing Collectors: Tallest in Each City 15-31
groupingBy: Additional Processing with entrySet() 15-32
Agenda 15-33

Nested Values	15-34
Data Organization of ComplexSalesTxn	15-35
Displaying Nested Values: Listing Line Items in Each Transaction	15-36
Displaying Nested Values: Listing Line items	15-37
Displaying Nested Values: Grouping LineItem elements	15-38
groupBy Examples for ComplexSalesTxn	15-39
Group Items by Salesperson	15-40
Agenda	15-42
Complex Custom Collectors	15-43
The collect Method: Using a Custom Collector	15-44
Creating a Custom Collector: Methods to Implement	15-45
Custom Example MyCustomCollector	15-46
Finisher Example MyCustomCollector	15-47
A More Complex Collector	15-48
A More Complex Collector CustomGroupingBy	15-49
Summary	15-51
Practice 15: Overview	15-52

16 Creating Custom Streams

Objectives	16-2
Topics	16-3
Performance: Intuition and Measurement	16-4
Parallel Versus Sequential Example	16-5
Spliterator	16-6
Decomposition with trySplit()	16-8
Integration with Streams	16-9
Modifying LongStream Spliterator	16-10
Spliterator TestSpliterator	16-11
Using TestSpliterator	16-12
Topics	16-13
Creating a Custom Spliterator	16-14
Is a Custom Spliterator Needed for a Game Engine?	16-15
A Tic-Tac-Toe Engine Using the map Method of Stream	16-16
A Custom Spliterator Example for a Custom Collection	16-17
Custom N-ary Tree	16-18
Possible Implementation for NaryTreeSpliterator	16-19
Stream<Node> in Use	16-20
Implementing Parallel Processing	16-21
Stream<Node> in Use	16-22
Summary of NTreeAsListSpliterator	16-23

Summary 16-24
Practice 16: Overview 16-25

17 Java I/O Fundamentals and File I/O (NIO.2)

Objectives 17-2
Java I/O Basics 17-3
I/O Streams 17-4
I/O Application 17-5
Data Within Streams 17-6
Byte Stream InputStream Methods 17-7
Byte Stream: Example 17-8
Character Stream Methods 17-9
Character Stream: Example 17-10
I/O Stream Chaining 17-11
Chained Streams: Example 17-12
Console I/O 17-13
Writing to Standard Output 17-14
Reading from Standard Input 17-15
Channel I/O 17-16
Persistence 17-17
Serialization and Object Graphs 17-18
Transient Fields and Objects 17-19
Transient: Example 17-20
Serial Version UID 17-21
Serialization: Example 17-22
Writing and Reading an Object Stream 17-23
Serialization Methods 17-24
readObject: Example 17-25
New File I/O API (NIO.2) 17-26
Limitations of java.io.File 17-27
File Systems, Paths, Files 17-28
Relative Path Versus Absolute Path 17-29
Java NIO.2 Concepts 17-30
Path Interface 17-31
Path Interface Features 17-32
Path: Example 17-33
Removing Redundancies from a Path 17-34
Creating a Subpath 17-35
Joining Two Paths 17-36
Symbolic Links 17-37
Working with Links 17-38

File Operations 17-39
BufferedReader File Stream 17-40
NIO File Stream 17-41
Read File into ArrayList 17-42
Managing Metadata 17-43
Summary 17-44
Quiz 17-45
Practice 17: Overview 17-50

18 Secure Coding Guidelines

Objectives 18-2
Java SE Security Overview 18-3
Secure Coding Guidelines 18-5
Vulnerabilities 18-6
Secure Coding Antipatterns 18-7
Antipatterns in Java 18-8
Fundamentals 18-9
Fundamentals: Why Should I Care? 18-16
Denial of Service 18-17
Confidential Information 18-20
Injection and Inclusion 18-24
Accessibility and Extensibility 18-26
Input Validation 18-31
Mutability 18-32
Object Construction 18-35
Serialization and Deserialization 18-38
Summary 18-40
Resources 18-41
Practice 18: Overview 18-42
Quiz 18-43

19 Building Database Applications with JDBC

Objectives 19-2
What Is the JDBC API? 19-3
What Is JDBC Driver? 19-4
Connecting to a Database 19-5
Obtaining a JDBC Driver 19-6
Register the JDBC driver with the DriverManager 19-7
Constructing a Connection URL 19-8
Establishing a Connection 19-9
Using the JDBC API 19-10

Key JDBC API Components	19-11
Writing Queries and Getting Results	19-12
Using a ResultSet Object	19-13
CRUD Operations Using JDBC API: Retrieve	19-14
CRUD Operations Using JDBC: Retrieve	19-15
CRUD Operations Using JDBC API: Create	19-16
CRUD Operations Using JDBC API: Update	19-17
CRUD Operations Using JDBC API: Delete	19-18
SQLException Class	19-19
Closing JDBC Objects	19-20
try-with-resources Construct	19-21
Using PreparedStatement	19-22
Using PreparedStatement: Setting Parameters	19-23
Executing PreparedStatement	19-24
PreparedStatement:Using a Loop to Set Values	19-25
Using CallableStatement	19-26
Summary	19-27
Practice 19: Overview	19-28
Quiz	19-29

20 Localization

Objectives	20-2
Why Localize?	20-3
A Sample Application	20-4
Locale	20-5
Properties	20-6
Loading and Using a Properties File	20-7
Loading Properties from the Command Line	20-8
Resource Bundle	20-9
Resource Bundle File	20-10
Sample Resource Bundle Files	20-11
Initializing the Sample Application	20-12
Sample Application: Main Loop	20-13
The printMenu Method	20-14
Changing the Locale	20-15
Sample Interface with French	20-16
Format Date and Currency	20-17
Displaying Currency	20-18
Formatting Currency with NumberFormat	20-19
Displaying Dates	20-20
Displaying Dates with DateTimeFormatter	20-21

Format Styles 20-22
Summary 20-23
Practice 20: Overview 20-24
Quiz 20-25

A Annotations

Objectives A-2
Topics A-3
Scenario A-4
@FunctionalInterface Annotation A-5
Annotation Characteristics A-6
@Override Annotation A-7
@Deprecated Annotation A-8
@Deprecated Annotation Recommendations and Options A-9
@SuppressWarnings Annotation A-10
@SafeVarargs Annotation A-11
Topics A-12
Examples from the Deprecated Annotation A-13
@Documented Meta Annotation Effects A-14
Topics A-15
Scenario A-16
Solution: Write a Custom Annotation A-17
Applying a Custom Annotation A-18
Reading Annotation Elements Through Reflection A-19
Other Details A-20
Inheriting an Annotation A-21
Reading an Inheriting an Annotation Through Reflection A-22
Repeating an Annotation A-23
Repeating Annotation Example A-24
Reading a Repeatable Annotation Through Reflection A-25
Topics A-26
Frameworks A-27
@NonNull Type Annotation and Pluggable Type Systems A-28
Summary A-29

B Security Survey

Objectives B-2
About This Lesson B-3
Topics B-4
Denial of Service (DoS) Attack B-5
Sockets B-6

Other DoS Examples	B-7
Topics	B-8
Enemies Target Confidential Information	B-9
Purge Sensitive Information from Exceptions	B-10
Do Not Log Highly Sensitive Information	B-11
Topics	B-12
Validate Inputs	B-13
Numbers That Make Programs Go Awry	B-14
Directory Traversal Attacks with ../	B-15
SQL Injection Through Dynamic SQL	B-16
Safer SQL	B-17
XML Inclusion	B-18
The Problem with XML Entities	B-19
Failure to Verify Bytecode	B-20
Topics	B-21
Isolate Unrelated Code	B-22
Stronger Encapsulation with Modules	B-23
Stronger Encapsulation Against Reflection	B-24
Limit Extensibility	B-25
Beware of Superclass Changes	B-26
Problem: Vulnerable Object Fields	B-27
Solution: Create Copies of Mutable or Subclassable Input Values	B-28
File Security Bug Reports	B-29
Topics	B-30
Serialization and Deserialization	B-31
Problem: Fields Are Accessible After Serialization	B-32
Solution 3a: Implement writeObject and readObject Methods	B-33
Solution 3b: Implement writeObject with PutField, and readObject with GetField	B-34
Solution 4: Implement a Serialization Proxy Pattern	B-35
Deserialize Cautiously	B-36
Summary	B-37

Adolfo De+la+Rosa (adolfodelarosa2012@gmail.com) has a
non-transferable license to use this Student Guide.