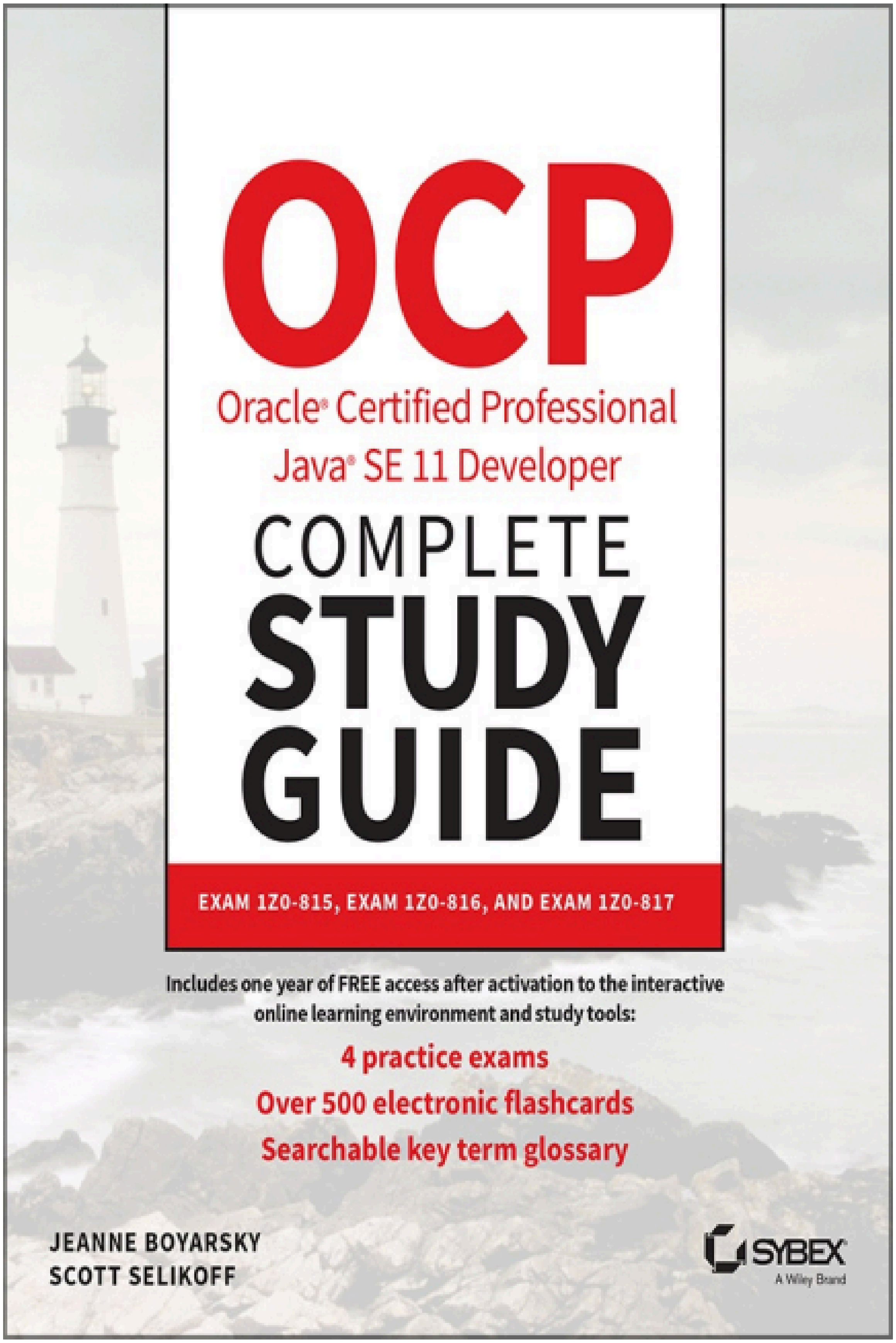
OCP
Oracle® Certified Professional
Java® SE 11 Developer
COMPLETE
STUDY
GUIDE
EXAM 1Z0-815, EXAM 1Z0-816, AND EXAM 1Z0-817
Includes one year of FREE access after activation to the interactive online learning environment and study tools:
4 practice exams
Over 500 electronic flashcards
Searchable key term glossary
JEANNE BOYARSKY
SCOTT SELIKOFF
SYBEX
A Wiley Brand

Table of Contents

List of Tables

List of Illustrations

OCP®

Oracle Certified Professional®

Java SE 11 Developer

Complete Study Guide Exam 1Z0-815, Exam 1Z0-816, and Exam 1Z0-817

Jeanne Boyarsky

Scott Selikoff

SYBEX A Wiley Brand

Published simultaneously in Canada and the United Kingdom

ISBN: 978-1-119-61913-0
ISBN: 978-1-119-61915-4 (ebk.)
ISBN: 978-1-119-61914-7 (ebk.)

Library of Congress Control Number: 2020938721

Happy 20th anniversary to NYC FIRST and StuyPulse FRC

Team 694. —Jeanne

For my daughter, Olivia, your determination and strength of heart are one of a kind. Your smile brightens even the darkest days. May your life be filled with happiness and love.

—Scott

Acknowledgments

Jeanne and Scott would like to thank numerous individuals for their contribution to this book. Thank you to Kathryn Duggan for guiding us through the process and making the book better in so many ways. Thank you to Janeice DelVecchio for being our technical editor as we wrote this book. Janeice pointed out many subtle errors in addition to the big ones. And thank you to Elena Felder for being our technical proofreader and finding the errors that we managed to sneak by Janeice. This book also wouldn't be possible without many people at Wiley, including Kenyon Brown, Pete Gaughan, Christine O'Connor, Barath Kumar Rajasekaran, Kim Wimpsett, Johnna VanHoose Dinse, and so many others.

Jeanne would personally like to thank Chris Kreussling for knowing more than a decade ago that she would someday write a book. He was a great mentor for many years and definitely shaped her career. Sibon Barman was helpful in getting feedback on the modules chapter, and Susanta Chattopadhyay provided real‐life use cases for both service locator and serialization. Stuart Dabbs Halloway's 2001 book provided examples of serialPeristentFields. Scott was a great co‐author, improving everything Jeanne wrote while writing his own chapters. A big thank you to everyone at CodeRanch.com who asked and responded to questions and comments about our books. Finally, Jeanne would like to thank all of the new programmers at CodeRanch.com and FIRST robotics teams FRC 694, FTC 310, and FTC 479 for the constant reminders of how new programmers think.

Scott could not have reached this point without his wife, Patti, and family, whose love and support makes this book possible. He would like to thank his twin daughters, Olivia and Sophia, and youngest daughter, Elysia, for
their patience and understanding especially when it was “time for Daddy to work in his office!” Scott would like to extend his gratitude to his wonderfully patient co‐author, Jeanne, on this, their fifth book. He doesn't know how she puts up with him, but he's glad she does and thrilled at the quality of books we produce. A big thanks to Matt Dalen, who has been a great friend, sounding board, and caring father to Olivia, Adeline, and newborn Henry. Finally, Scott would like to thank his mother and retired
teacher, Barbara Selikoff, for teaching him the value of education, and his father, Mark Selikoff, for instilling in him the benefits of working hard.

We'd both like to thank Marcus Biel for providing a European's take on our localization content. Last but not least, both Jeanne and Scott would like to give a big thank you to the readers of all our books. Hearing from all of you who enjoyed the book and passed the exam is a great feeling. We'd also like to thank those who pointed out errors and made suggestions for improvements in the 1Z0‐815 Java 11 book. As of May 2020, the top two were Nikolai Vinoku and Edmond Yong. Also, an honorable mention to Jakub

Chrobak. Finally, thank you to Atanas Gegov for submitting a pull request to improve the 1Z0-815 modules examples readme.

About the Authors

Jeanne Boyarsky was selected as a Java Champion in 2019. She has worked as a Java developer for more than 18 years at a bank in New York City where she develops, mentors, and conducts training. Besides being a senior moderator at CodeRanch.com in her free time, she works on the forum code base. Jeanne also mentors the programming division of a FIRST robotics team where she works with students just getting started with Java. She also speaks at several conferences each year.

Jeanne got her Bachelor of Arts degree in 2002 and her Master in Computer Information Technology degree in 2005. She enjoyed getting her Master's degree in an online program while working full-time. This was before online education was cool! Jeanne is also a Distinguished Toastmaster and a Scrum Master. You can find out more about Jeanne at www.jeanneboyarsky.com or follow her on Twitter @JeanneBoyarsky.

Scott Selikoff is a professional software consultant, author, and owner of Selikoff Solutions, LLC, which provides software development solutions to businesses in the tri-state New York City area. Skilled in a plethora of software languages and platforms, Scott specializes in full- stack database-driven systems, cloud-based applications, microservice architectures, and service-oriented architectures.

A native of Toms River, New Jersey, Scott achieved his Bachelor of Arts degree from Cornell University in Mathematics and Computer Science in 2002, after three years of study. In 2003, he received his Master of Engineering degree in Computer Science, also from Cornell University.

As someone with a deep love of education, Scott has always enjoyed teaching others new concepts. He's given lectures at Cornell University and Rutgers University, as well as conferences including Oracle Code One and The Server Side Java Symposium. Scott lives in New Jersey with his loving wife, Patti; three amazing daughters, twins Olivia and Sophia and little Elysia; and two very playful dogs, Webby and Georgette. You can find out more about Scott at www.linkedin.com/in/selikoff or follow him on Twitter @ScottSelikoff. Jeanne and Scott are both moderators on the CodeRanch.com forums and can be reached there for question and comments. They also co-author a technical blog called Down Home Country

Coding at www.selikoff.net.

In addition to this book, Jeanne and Scott are also authors of the following best-selling Java 8 certification books: *OCA Oracle Certified Associate Java SE 8 Programmer I Study Guide* (Sybex, 2015) and *OCP Oracle Certified Professional Java SE 8 Programmer II Study Guide* (Sybex, 2016). These two books have been combined into the single release: *OCA/OCP Java SE 8 Programmer Certification Kit: Exam 1Z0-808 and Exam 1Z0-809* (Sybex 2016). They have also written a book of practice test questions for the Java 8 certification exams: *OCA/OCP Java SE 8 Programmer Practice Tests* (Sybex, 2017). Their most recent books are *OCP Oracle Certified Professional Java SE 11 Programmer I Study Guide: Exam 1Z0-815* (Sybex, 2019) and *OCP Oracle Certified Professional Java SE 11 Programmer II Study Guide: Exam 1Z0-816* (Sybex, 2020).

Introduction

This book is for those looking to obtain an Oracle Certified Professional Java SE 11 Developer or Java Foundations Certified Junior Associate title. This book is also for those looking to gain a deeper understanding and appreciation of Java. Not only do we want you to pass your exams, but we want to help you to improve yourself and become a better professional software developer.

The book provides detailed preparation for the following Oracle certifications exams:

1Z0-815 Exam: Java SE 11 Programmer I The Programmer I Exam covers a wide variety of core topics in Java 11 including classes, interfaces, lambda expressions, operators, decision constructs, basic collections, and modules. These topics form the foundation of most Java applications.

1Z0-816 Exam: Java SE 11 Programmer II The Programmer II Exam delves into greater detail on select topics in Java 11 including streams, modular applications, generics, advanced collections, I/O and NIO.2, concurrency, annotations, and security.

IZ0-817 Exam: Upgrade OCP Java 6, 7 & 8 to Java SE 11 Developer The Upgrade Exam is meant for those who hold an existing OCP certification to be able to obtain the Java 11 OCP certification title with a single exam. It contains a selection of Java 11 topics from both the Programmer I and Programmer II exams.

1Z0-811 Exam: Java Foundations The Foundations Exam a junior level certification exam that contains a variety of introductory and basic Java 8 topics. It is not meant for existing Java professionals, but rather those who use Java infrequently in their job or don't want to dive as deep into Java.

In the introduction, we start by covering important information about the various exams. Depending on your certification history, you may have a choice of which exam you can take. We then move on to information
about how this book is structured. Finally, we conclude with two assessment tests so you can see how much studying lies ahead of you.

Understanding the Exam

At the end of the day, the exam is a list of questions. The more you know about the structure of the exam, the better you are likely to do. For example, knowing how many questions the exam contains allows you to manage your progress and time remaining better. In this section, we discuss the details of the exam, along with some history of previous certification exams.

BROADER OBJECTIVES

In previous certification exams, the list of exam objectives tended to include specific topics, classes, and APIs that you needed to know for the exam. For example, take a look at an objective for the 1Z0- 809 (OCP 8) exam:

> Use BufferedReader, BufferedWriter, File, FileReader, FileWriter, FileInputStream, FileOutputStream, ObjectOutputStream, ObjectInputStream, and PrintWriter in the java.io package.

Now compare it with the equivalent objective for the 1Z0-816 (OCP 11) exam:

> Use I/O Streams to read and write files

Notice the difference? The older version is more detailed and describes specific classes you will need to understand. The newer version is a lot vaguer. It also gives the exam writers a lot more freedom to insert a new feature without having to update the list of objectives.

So how do you know what to study? By reading this study guide of course! We've spent years studying the certification exams, in all of their forms, and have carefully cultivated topics, material, and practice questions that we are confident can lead to successfully passing the exam.

CHOOSING WHICH EXAM TO TAKE

Java is now 25 years old, celebrating being “born” in 1995. As with anything 25 years old, there is a good amount of history and variation between different versions of Java. Over the years, the certification exams have changed to cover different topics. The names of the exams have even changed. This book covers the Java 11 exam.

Those with more recent certifications might remember that Oracle released two exams each for Java 7 and Java 8. The first exam tended to be easier, and completing it granted you the title of Oracle Certified Associate (OCA). The second exam was a lot more difficult, with much longer questions, and completing it granted you the title of Oracle Certified Professional (OCP).

Oracle did not release an exam for Java 9 or Java 10, probably because neither of these is a Long Term Support (LTS) release. With Java 11, Oracle decided to discontinue both the OCA certification and its associated exam. You still have to take two exams to earn an OCP title. Both are more difficult than the old OCA exams. The difference is that now you do not obtain a certification title from completing the first exam.

Figure I.1 shows these past and current Java certifications. This image is helpful if you run into material online that references older exams. It is also helpful if you have an older certification and are trying to determine where it fits in.

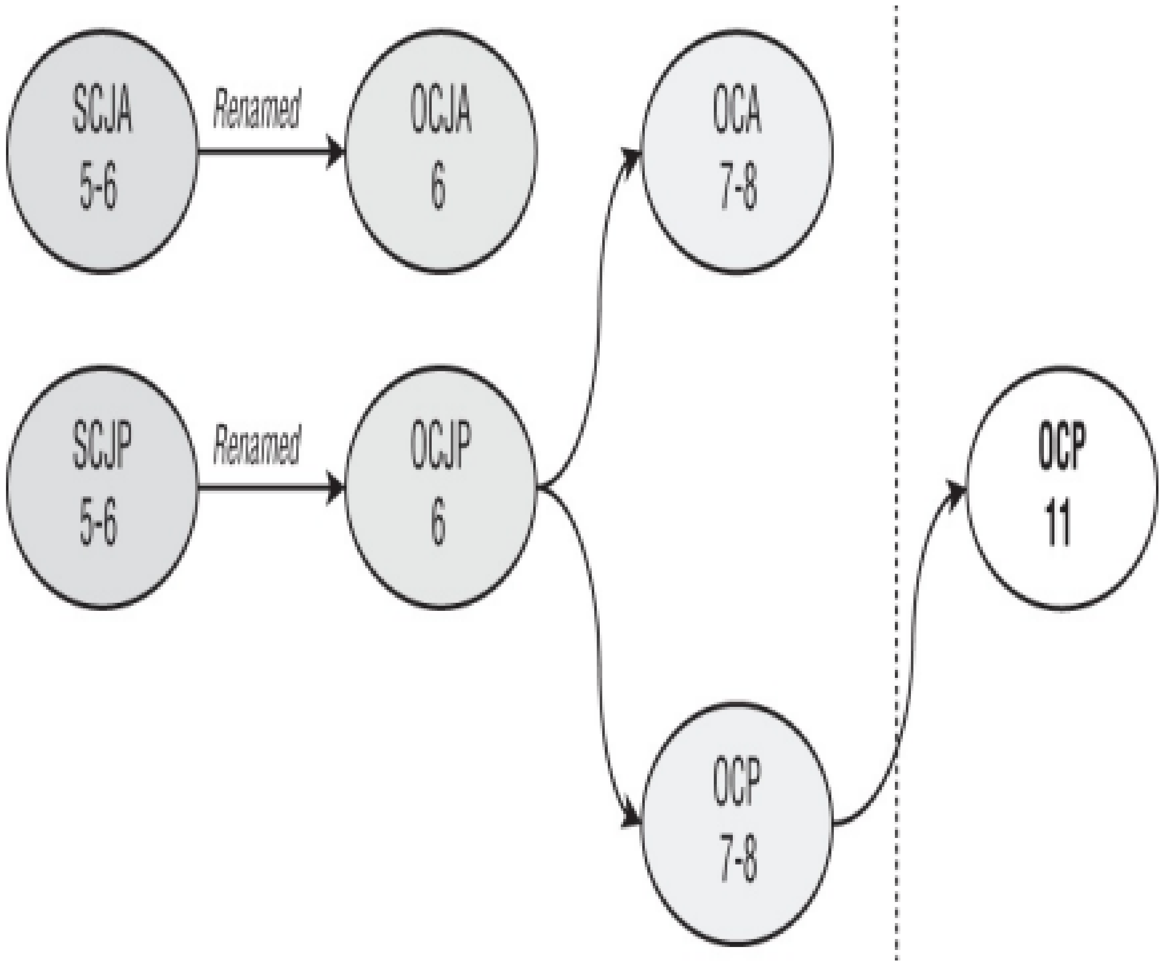

FIGURE I.1 Past and current Java certifications

Figure I.2 shows the exams you need to take in order to earn the latest Java certification if you are new to certification.

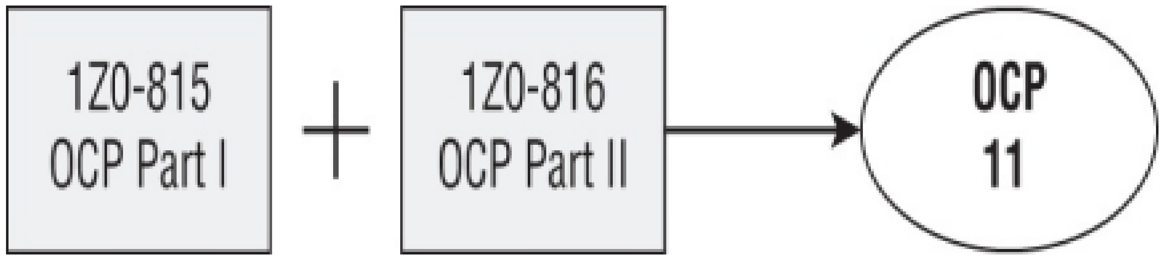

FIGURE I.2 Latest Java certification exams

For those who already hold a Java certification, Figure I.3 shows common scenarios for which exam(s) you should target.

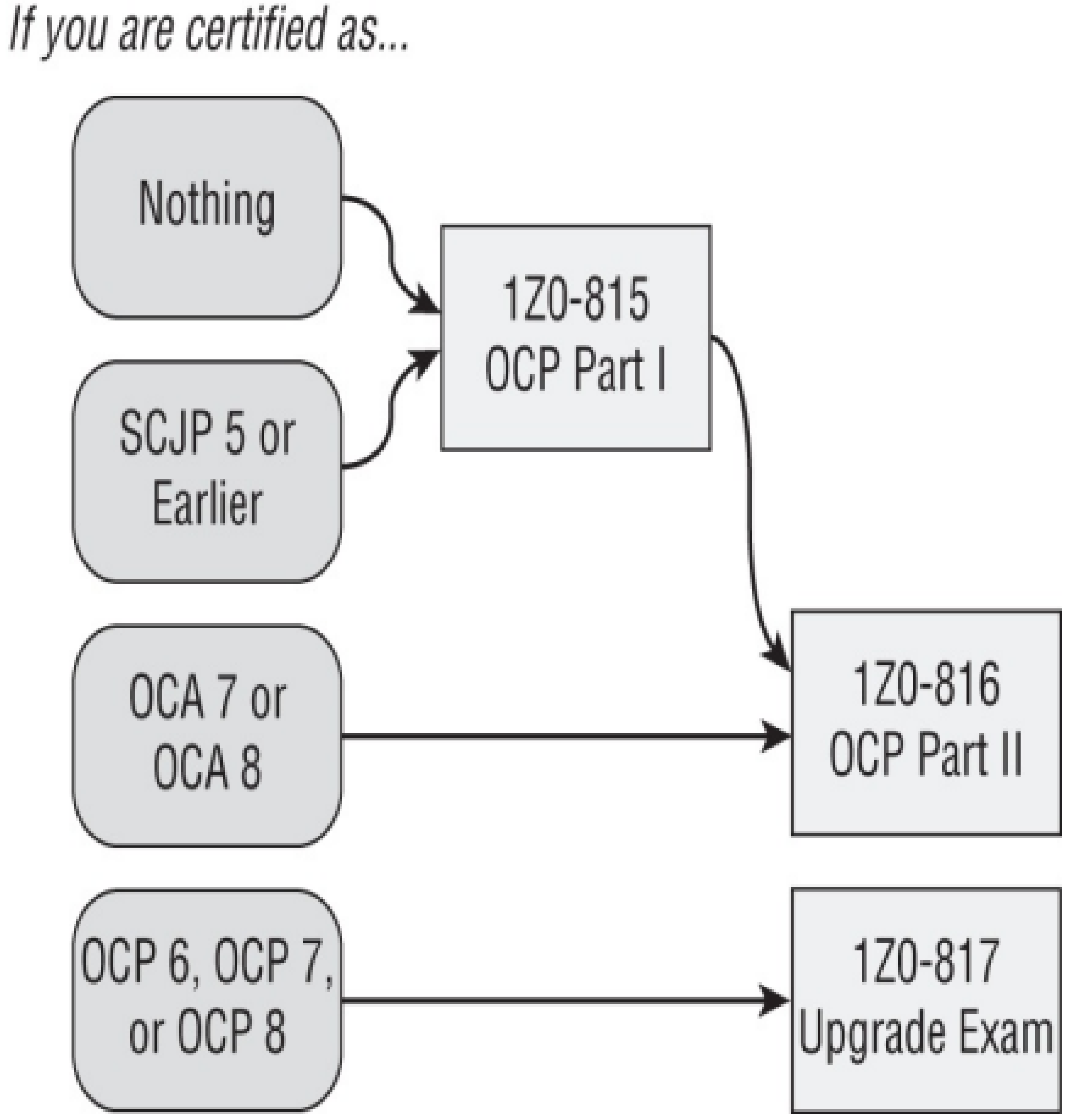

FIGURE I.3 Exam prerequisites

In a nutshell, you can take the 1Z0-816 exam if you passed the 1Z0-815 exam or hold the OCA 7 or 8 title. Oracle's goal here is to help people get to Java 11 OCP certification if they are halfway through the journey to OCP certification. Similarly, those with an OCP certification can take the 1Z0-817 upgrade exam to get to Java 11 OCP with one exam. Those with an older certification will have to start over and take the 1Z0-815 exam.

There are also two edge cases. Those who passed the OCA 6 exam must still take the 1Z0-815 exam. The OCA 6 exam covered far less material than the OCA 7 or 8.
Additionally, those who passed the OCP 7 or 8 exam but never received the OCP title because they didn't pass the OCA exam, need to take the 1Z0-815 exam. After that, you have a choice of the 1Z0-816 exam or 1Z0- 817 exam. We recommend reading the exam objectives for both and picking the one that tests the topics that you know better.

TIP

If you're not sure which exam you should take, you can post questions on CodeRanch.com and the community will be happy to help. You might even get a response from Jeanne or Scott!

TAKING THE UPGRADE EXAM

The chapters of this book are structured for those taking the 1Z0-815 Programmer I and 1Z0-816 Programmer II exams. Those taking the 1Z0- 817 Upgrade Exam can also rely on this book to prepare for the exam, and you don't need to read all 22 chapters!

While we think every chapter is worth reading, the following is a list of chapters that you should focus on if you are preparing for the 1Z0-817 exam and the order you should read them:

Chapter 2, "Java Building Blocks"

Chapter 11, "Modules"

Chapter 12, "Java Fundamentals"

Chapter 14, "Generics and Collections" (Lambda expressions and method references)

Chapter 15, "Functional Programming"

Chapter 16, "Exceptions, Assertions, and Localization"

Chapter 17, "Modular Applications"

Chapter 18, "Concurrency"

Chapter 20, "NIO.2"

The 1Z0-817 exam is cumulative, which means material from the 1Z0-815 exam is fair game. For instance, the 1Z0-817 exam does not have any objectives on while and for loops, but they are certainly likely to appear in questions. In other words, if it's been awhile since you took the previous OCP exam, we recommend reading all of the chapters in Part I of this book.

We've included a mapping of all of the upgrade exam objectives and their associated chapters in the "Objective Maps" section of this introduction.

CHANGES TO THE EXAM

Table I.1 shows the information about the exams at the time of publishing. **TABLE I.1** Exam information

Exam	Len gt h	# Ques tio ns	Passi ng Scor e
1Z0-815 Java Programmer I	3 hours	80	63%
1Z0-816 Java Programmer II	3 hours	80	63%
1Z0-817 Upgrade OCP Java 6, 7 & 8 to Java SE 11 Developer	3 hours	80	61%
1Z0-811 Java Foundations	2.5 hours	75	65%

Oracle has a tendency to fiddle with the length of the exam and the passing score once it comes out. Oracle also likes to “tweak” the exam topics over time. It wouldn't be a surprise for Oracle to make minor changes to the exam objectives, the number of questions, or the passing score after this book goes to print.

If there are any changes to the exam after this book is published, we will note them on the book page of our blog:

www.selikoff.net/ocp11-complete/

EXAM QUESTIONS

EXAM QUESTIONS

Each exam consists entirely of multiple choice questions. There are between four and seven possible answers. If a question has more than one answer, the question specifically states exactly how many correct answers there are. This book does not do that. We say “Choose all that apply” to make the questions harder. This means the questions in this book are generally harder than those on the exam. The idea is to give you more practice so you can spot the correct answer more easily on the real exam.

If you read about older versions of the exams online, you might see references to drag-and-drop questions. These questions had you do a puzzle on how to complete a piece of code. Luckily these are no longer on any of the exams.

Many of the questions on each exam are code snippets rather than full classes. Saving space by not including imports and/or class definitions leaves room for lots of other code. For example, it is common to come across classes on the exam with portions omitted, like so:

```
public class Zoo {
    String name;
    // Getters/Setters/Constructors omitted
}
```

In this case, you would assume methods like getName() and setName(), as well as related constructors exist. For instance, we would expect this code to compile:

```
var name = new Zoo("Java Zoo").getName();
```

OUT-OF-SCOPE MATERIAL

When you take an exam, you may see some questions that appear to be out of scope. *Don't panic!* Often, these questions do not require knowing anything about the topic to answer the question. For example, after reading this book you should be able to spot that the following does not compile, even if you've never heard of LocalDate and ChronoUnit:

```
final LocalDate holiday = LocalDate.now();
holiday = LocalDate.now().plus(5,ChronoUnit.HOURS);
```

The classes and

enums used in this question are not in scope for the exam,

but the reason it does not compile is in scope. In particular, you should know that you cannot reassign a variable marked final.

See, not so scary is it? Expect to see at least a few structures on the exam that you are not familiar with. If they aren't part of your exam preparation material, then you don't need to understand them to answer the question.

QUESTION TOPIC TIPS

The following list of topics is meant to give you an idea of the types of questions and oddities that you might come across on the exam. Being aware of these categories of such questions will help you get a higher score on an exam.

Questions with Extra Information Provided Imagine the question includes a statement that XMLParseException is a checked exception. It's fine if you don't know what an XMLParseException is or what XML is for that matter. (If you are wondering, it is a format for data.) This question is a gift. You know the question is about checked and unchecked exceptions.

Questions with Embedded Questions To answer some questions on the exam, you may have to actually answer two or three subquestions. For example, the question may contain two blank lines and the question may ask you to choose the two answers that fill in each blank. In some cases, the two answer choices are not related, which means you're really answering multiple questions, not just one! These questions are among the most difficult and time-consuming on the exam because they contain multiple, often independent, questions to answer. Unfortunately, the exam does not give partial credit, so take care when answering questions like these.

Questions with Unfamiliar APIs If you see a class or method that wasn't covered in this book, assume it works as you would expect. Some of these APIs you might come across, such as LocalDate, were on the Java 8 exam and are not part of the Java 11 exams. Assume that the part of the code using that API is correct and look very hard for other errors.

Questions with Made-Up or Incorrect Concepts In the context of a word problem, the exam may bring up a term or concept that does not make any sense such as saying an interface inherits from a class, which is not a correct statement. In other cases, they may use a keyword that does not exist in Java, like struct. For these, you just have to read them
carefully and recognize when the exam is using invalid terminology.

Questions That Are Really Out of Scope When introducing new questions, Oracle includes them as unscored questions at first. This allows them to see how real exam takers do without impacting your score. You will still receive the number of questions as the exam lists. However, a few of them may not count. These unscored questions may contain out-of- scope material or even errors. They will not be marked as unscored so you still have to do your best to answer them. Follow the previous advice to assume that anything you haven't seen before is correct. That will cover you if the question is being counted!

Reading This Book

It might help to have some idea about how this book has been written. This section contains details about some of the common structures and features you will find in this book, where to go for additional help, and how to obtain bonus material for this book.

WHO SHOULD BUY THIS BOOK

If you want to obtain the OCP 11 Java programmer certification, this book is definitely for you. If you want to acquire a solid foundation in Java and your goal is to prepare for the exam, then this book is also for you. You'll find clear explanations of the concepts you need to grasp and plenty of help to achieve the high level of professional competency you need in order to succeed in your chosen field.

This book is intended to be understandable to anyone who has a tiny bit of Java knowledge. If you've never read a Java book before, we recommend starting with a book that teaches programming from the beginning and then returning to this study guide.

This book is for anyone from high school students to those beginning their programming journey to experienced professionals who need a review for the certification.

HOW THIS BOOK IS ORGANIZED

This book is divided into two parts consisting of 11 chapters each, plus supplementary information: an online glossary, this introduction, and
multiple bonus exams. You might have noticed that there are more than 11 objectives for each exam. We organized what you need to know to make it easy to learn and remember. Each chapter begins with a list of the objectives that are covered in that chapter.

Part I: Exam 1Z0-815, OCP Java SE 11 Programmer I:

Chapter 1: Welcome to Java describes the basics of Java such as how to run a program. It also includes the benefits of Java and key terminology.

Chapter 2: Java Building Blocks focuses on variables such as primitives and object data types and scoping variables. It also discusses garbage collection.

Chapter 3: Operators explains operations with variables. It also talks about casting and the precedence of operators.

Chapter 4: Making Decisions covers on core logical constructs such as conditionals and loops.

Chapter 5: Core Java APIs introduces you to String, StringBuilder, array, and various types.

Chapter 6: Lambdas and Functional Interfaces shows how to use lambdas and four key functional interfaces. The focus is implementing and calling Predicate, Consumer, Supplier, and Comparator.

Chapter 7: Methods and Encapsulation explains how to write methods. It also shows the four access modifiers.

Chapter 8: Class Design covers constructors and superclasses. It also includes method overriding.

Chapter 9: Advanced Class Design adds interfaces and abstract classes. It also introduces inner classes.

Chapter 10: Exceptions shows the different types of exception classes and how to use them. It also includes different uses of try statements.

Chapter 11: Modules details the benefits of the new module feature. It shows how to compile and run module programs from the command line.

Part II: Exam 1Z0-816, OCP Java SE 11 Programmer IIExam 1Z0-817, Upgrade OCP Java SE 11:

Chapter 12: Java Fundamentals covers core Java topics including enums, the final modifier, inner classes, and interfaces. There are now many types of functional interface methods that you need to know for the exam. It also includes an introduction to creating functional interfaces and lambda expressions.

Chapter 13: Annotations describes how to define and apply your own custom annotations, as well as how to use the common built-in ones.

Chapter 14: Generics and Collections goes beyond the basics and demonstrates method references, generics with wildcards, and Collections. The Collections portion covers many common interfaces, classes, and methods that are useful for the exam and in every day software development.

Chapter 15: Functional Programming explains lambdas and stream pipelines in detail. It also covers the built-in functional interfaces and the Optional class. If you want to become skilled at creating streams, read this chapter more than once!

Chapter 16: Exceptions, Assertions, and Localization shows advanced exception handling topics including creating custom exceptions, try-with- resources statements, and suppressed exceptions. It also covers how to use assertions to validate your program. It concludes with localization and formatting, which allows your program to gracefully support multiple countries or languages.

Chapter 17: Modular Applications shows advanced modularization concepts including services and how to migrate an application to a modular infrastructure.

Chapter 18: Concurrency introduces the concept of thread management, and teaches you how to build multi-threaded programs using the concurrency API and parallel streams.

Chapter 19: I/O introduces you to managing files and directories using the java.io API. It covers a number of I/O stream classes, teaches you how to serialize data, and shows how to interact with a user.

Chapter 20: NIO.2 shows you to manage files and directories using the newer NIO.2 API. It includes techniques for using streams to traverse and search the file system.

Chapter 21: JDBC provides the basics of working with databases in Java including working with stored procedures.

Chapter 22: Security describes how to securely build your program and protect against common malicious attacks.

At the end of each chapter, you'll find a few elements you can use to prepare for the exam:

Summary This section reviews the most important topics that were covered in the chapter and serves as a good review.

Exam Essentials This section summarizes highlights that were covered in the chapter. You should be able to convey the information described.

Review Questions Each chapter concludes with at least 20 review questions. You should answer these questions and check your answers against the ones provided in the Appendix. If you can't answer at least 80 percent of these questions correctly, go back and review the chapter, or at least those sections that seem to be giving you difficulty.

WARNING

The review questions, assessment tests, and other testing elements included in this book are *not* derived from the real exam questions, so don't memorize the answers to these questions and assume that
doing so will enable you to pass the exam. You should focus on understanding the topic, as described in the text of the book. This will let you answer the questions provided with this book *and* pass the exam. Learning the underlying topic is also the approach that will serve you best in the workplace—the ultimate goal of a certification.

To get the most out of this book, you should read each chapter from start to finish before going to the chapter-end elements. They are most useful for checking and reinforcing your understanding. Even if you're already
familiar with a topic, you should skim the chapter. There are a number of subtleties to Java that you could easily not encounter even when working with Java for years.

CONVENTIONS USED IN THIS BOOK

CONVENTIONS USED IN THIS BOOK

This book uses certain typographic styles to help you quickly identify important information and to avoid confusion over the meaning of words such as on-screen prompts. In particular, look for the following styles:

Italicized text indicates key terms that are described at length for the first time in a chapter. (Italics are also used for emphasis.)

A monospaced font indicates code or command-line

text. *Italicized monospaced text* indicates a variable.

In addition to these text conventions, which can apply to individual words or entire paragraphs, a few conventions highlight segments of text.

A tip is something to call particular attention to an aspect of working with a language feature or API.

A note indicates information that's useful or interesting. It is often something to pay special attention to for the exam.

SIDEBARS

A sidebar is like a note but longer. The information in a sidebar is useful, but it doesn't fit into the main flow of the text.

Real World Scenario

A real-world scenario is a type of sidebar that describes a task or an example that's particularly grounded in the real world. This is something that is useful in the real world, but is not going to show
up on the exam.

GETTING HELP

Both of the authors are moderators at CodeRanch.com. This site is a quite large and active programming forum that is friendly toward Java beginners. It has a forum just for this exam called Programmer Certification. It also has a forum called Beginning Java for non-exam- specific questions. As you read the book, feel free to ask your questions in either of those forums. It could be you are having trouble compiling a class or that you are just plain confused about something. You'll get an answer from a knowledgeable Java programmer. It might even be one of us.

INTERACTIVE ONLINE LEARNING ENVIRONMENT

AND TEST BANK

We've put together some really great online tools to help you pass the exams. The interactive online learning environment that accompanies this study guide provides a test bank and study tools to help you prepare for the exam. By using these tools you can dramatically increase your chances of passing the exam on your first try.

The online test bank includes the following:

Practice Exams Many sample tests are provided throughout this book and online, including the assessment tests, which you'll find at the end of this introduction, and the chapter tests that include the review questions at the end of each chapter. In addition, there are four bonus practice exams. Use these questions to test your knowledge of the study guide material. The online test bank runs on multiple devices.

Flashcards The online text bank includes two sets of flashcards specifically written to hit you hard, so don't get discouraged if you don't ace your way through them at first! They're there to ensure that you're really ready for the exam. And no worries—armed with the review questions, practice exams, and flashcards, you'll be more than prepared when exam day comes! Questions are provided in digital flashcard format (a question followed by a single correct answer). You can use the flashcards to reinforce your learning and provide last-minute test prep
before the exam.

Resources A glossary of key terms from this book and their definitions is available as a fully searchable PDF.

To register and gain access to this interactive online learning environment, please visit this URL:

www.wiley.com/go/Sybextestprep

Preparing for the Exam

This section includes suggestions and recommendations for how you should prepare for the certification exam. If you're an experienced test taker or you've taken a certification test

before, most of this should be
common knowledge. For those who are taking the exam for the first time, don't worry! We'll present a number of tips and strategies to help you prepare for the exam.

CREATING A STUDY PLAN

Rome wasn't built in a day, so you shouldn't attempt to study for the exam in only one day. Even if you have been certified with a previous version of Java, the new test includes features and components unique to Java 9, 10, and 11 that are covered in this text.

Once you have decided to take the test, you should construct a study plan that fits with your schedule. We recommend that you set aside some amount of time each day, even if it's just a few minutes during lunch, to read or practice for the exam. The idea is to keep your momentum going throughout the exam preparation process. The more consistent you are in how you study, the better prepared you will be for the exam. Try to avoid taking a few days or weeks off from studying or you're likely to spend a lot of time relearning existing material instead of moving on to new material.

CREATING AND RUNNING THE CODE

Although some people can learn Java just by reading a textbook, that's not how we recommend that you study for a certification exam. We want you to be writing your own Java sample applications throughout this book so that you don't just learn the material, but that you understand the material as well. For example, it may not be obvious why the following line of code does not compile, but if you try to compile it yourself, the Java compiler will tell you the problem:

```
float value = 102.0; // DOES NOT COMPILE
```

NOTE

A lot of people post the question “Why does this code not compile?” on the CodeRanch.com forum. If you're stuck or just curious about a behavior in Java, we encourage you to post to the forum. There are a lot of nice people in the Java community standing by to help you.

SAMPLE TEST CLASS

Throughout this book, we present numerous code snippets and ask you whether they'll compile or not and what their output will be. You will place these snippets inside a simple Java application that starts, executes the code, and terminates. You can accomplish this by compiling and running a `public` class containing a `public static void main(String[] args)` method and adding the necessary import statements, such as the following:

```
// Add any necessary import statements here
public class TestClass {
    public static void main(String[] args) {
        // Add test code here

        // Add any print statements here
        System.out.println("Hello World!");
    }
}
```

This application isn't particularly interesting—it just outputs `Hello World!` and exits. That said, you could insert many of the code snippets presented in this book in the `main()` method to determine whether the
code compiles, as well as what the code outputs when it does compile.

Real World Scenario

IDE SOFTWARE

While studying for an exam, you should develop code using a text editor and command‐line Java compiler. Some of you may have prior experience with integrated development environments (IDEs), such as Eclipse, IntelliJ, or Visual Studio Code. An IDE is a software application that facilitates software development for computer programmers. Although such tools are extremely valuable in developing software, they can interfere with your ability to spot problems readily on an exam.

IDENTIFYING YOUR WEAKEST LINK

The review questions in each chapter are designed to help you hone in on those features of the Java language where you may be weak and that are required knowledge for the exam. For each chapter, you should note which questions you got wrong, understand why you got them wrong, and study those areas even more. After you've reread the chapter and written lots of code, you can do the review questions again. In fact, you can take the review questions over and over to reinforce your learning as long as you explain to yourself why it is correct.

"OVERSTUDYING" THE ONLINE PRACTICE EXAM

Although we recommend reading this book and writing your own sample applications multiple times, redoing the online practice exam over and over can have a negative impact in the long run. For example, some individuals study the practice exam so much that they end up memorizing the answers. In this scenario, they can easily become overconfident; that is, they can achieve perfect scores on the practice exams but may fail the actual exam.

UNDERSTANDING THE QUESTION

The majority of questions on each exam will contain code snippets and ask you to answer questions about them. For those items containing code snippets, the number-one question we recommend that you answer before attempting to solve the question is this:

Does the code compile?

It sounds simple, but many people dive into answering the question without checking whether the code actually compiles. If you can determine whether a particular set of code compiles and what line or lines cause it to not compile, answering the question often becomes easy.

APPLYING THE PROCESS OF ELIMINATION

Although you might not immediately know the correct answer to a question, if you can reduce the question from five answers to three, your odds of guessing the correct answer will be markedly improved. Moreover, if you can reduce a question from four answers to two, you'll double your chances of guessing the correct answer!

The exam software allows you to eliminate answer choices by right- clicking an answer choice, which causes the text to be struck through, as shown in the following example:

~~A. 123~~

B. Elephant

~~C. Vulture~~

D. The code does not compile due to line `n1`.

Even better, the exam software remembers which answer choices you have eliminated anytime you go back to the question. You can undo the crossed-out answer simply by right-clicking the choice again.

Sometimes you can eliminate answer choices quickly without reading the entire question. In some cases, you may even be able to solve the question based solely on the answer choices. If you come across such questions on the exam, consider it a gift. Can you correctly answer the following
question in which the application code has been left out?

5. Which line, when inserted independently at line `m1`, allows the code to compile?

   - Code Omitted -

1. public abstract final int swim();
2. public abstract void swim();
3. public abstract swim();
4. public abstract void swim() {}
5. public void swim() {}

Without reading the code or knowing what line `m1` is, we can actually eliminate three of the five answer choices. Options A, C, and D contain invalid declarations, leaving us with options B and E as the only possible correct answers.

SKIPPING DIFFICULT QUESTIONS

The exam software also includes an option to "mark" a question and review all marked questions at the end of the exam. If you are pressed for time, answer a question as best you can and then mark it to come back to later.

All questions are weighted equally, so spending 10 minutes answering five questions correctly is a lot better use of your time than spending 10 minutes on a single question. If you finish the exam early, you have the option of reviewing the marked questions, as well as all of the questions on the exam if you so choose.

BEING SUSPICIOUS OF STRONG WORDS

Many questions on each exam include answer choices with descriptive sentences rather than lines of code. When you see such questions, be wary of any answer choice that includes strong words such as “must,” “all,” or “cannot.” If you think about the complexities of programming languages, it is rare for a rule to have no exceptions or special cases. Therefore, if you are stuck between two answers and one of them uses “must” while the other uses “can” or “may,” you are better off picking the one with the weaker word since it is a more ambiguous statement.

USING THE PROVIDED WRITING MATERIAL

Depending on your particular testing center, you will be provided with a sheet of blank paper or a whiteboard to use to help you answer questions. In our experience, a whiteboard with a marker and an eraser are more commonly handed out. If you sit down and you are not provided with anything, make sure to ask for such materials.

After you have determined that the program does compile, it is time to understand what the program does! One of the most useful applications of writing material is tracking the state of primitive and reference variables. For example, let's say you encountered the following code snippet on a question about garbage collection:

```
Object o = new Turtle();
Mammal m = new Monkey();
Animal a = new Rabbit();
o = m;
```

In a situation like this, it can be helpful to draw a diagram of the current state of the variable references. As each reference variable changes which object it points to, you would erase or cross out the arrow between them and draw a new one to a different object.

Using the writing material to track state is also useful for complex questions that involve a loop, especially questions with embedded loops. For example, the value of a variable might change 5 or more times during a loop execution. You should make use of the provided writing material to improve your score.

While you cannot bring any outside material into an exam, you can write down material at the start of the exam. For example, if you have trouble remembering which functional interfaces take which generic arguments, then it might be helpful to draw a table at the start of the exam on the provided writing material. You can then use this information to answer multiple questions.

CHOOSING THE BEST ANSWER

Sometimes you read a question and immediately spot a compiler error that tells you exactly what the question is asking. Other times, though, you may stare at a method declaration for a couple of minutes and have no idea what the question is asking. While you might not know for sure which answer is correct in these situations, there are some test-taking tips that can improve the probability that you will pick the correct answer.

Unlike some other standardized tests, there's no penalty for answering a question incorrectly versus leaving it blank. If you're nearly out of time or you just can't decide on an answer, select a random answer and move on. If you've been able to eliminate even one answer, then your guess will be better than blind luck.

ANSWER ALL QUESTIONS!

You should set a hard stop at five minutes of time remaining on the exam to ensure that you've answered each and every question. Remember, if you fail to answer a question, you'll definitely get it wrong and lose points, but if you guess, there's at least a chance that you'll be correct. There's no harm in guessing!

When in doubt, we generally recommend picking a random answer that includes “Does not compile” if available, although which choice you select is not nearly as important as making sure that you do not leave any questions unanswered on the exam!

GETTING A GOOD NIGHT'S REST

Although a lot of people are inclined to cram as much material as they can in the hours leading up to an exam, most studies have shown that this is a poor test-taking strategy. The best thing we can recommend that you do before taking an exam is to get a good night's rest!

Given the length of each exam and number of questions, the exam can be quite draining, especially if this is your first time taking a certification exam. You might come in expecting to be done 30 minutes early, only to discover that you are only a quarter of the way through the exam with half
the time remaining. At some point, you may begin to panic, and it is in these moments that these test-taking skills are most important. Just remember to take a deep breath, stay calm, eliminate as many wrong answers as you can, and make sure to answer each and every question. It is for stressful moments like these that being well rested with a good night's sleep will be most beneficial!

Taking the Exam

So you've decided to take the exam? We hope so if you've bought this book! In this section, we discuss the process of scheduling and taking the exam, along with various options for each.

SCHEDULING THE EXAM

The exam is administered by Pearson VUE and can be taken at any Pearson VUE testing center. To find a testing center or register for the exam, go to:

www.pearsonvue.com

Next, search for *Oracle* as the exam provider. If you haven't been to the test center before, we recommend visiting in advance. Some testing centers are nice and professionally run. Others stick you in a closet with lots of people talking around you. You don't want to be taking the test with people complaining about their broken laptops nearby!

At this time, you can reschedule the exam without penalty until up to 24 hours before. This means you can register for a convenient time slot well in advance, knowing that you can delay if you aren't ready by that time. Rescheduling is easy and can be done completely on the Pearson VUE website. This may change, so check the rules before paying.

THE AT-HOME ONLINE OPTION

Oracle now offers online-proctored exams that can be taken in the comfort of your own home. You choose a specific date and time, like a proctored exam, and take it at your computer.

While this option may be appealing for a lot of people, especially if you live far away from a testing center, there are a number of restrictions.

Your session will be closely monitored by another individual from a remote location.

You must set up a camera and microphone, and they must be on for the entire exam. At the start, you will also need to turn the camera around the room to show your workspace to prove you are not in reach of exam material.

The exam software will also monitor your facial expressions and track eye movement. We've heard reports that it will warn you if you are looking away from the screen too much.

You must be alone in a completely isolated space for the duration of the test. If someone comes in during your test, your test will be invalidated.

You cannot have any papers, material, or items in your immediate vicinity.

Unlike exam centers that provide writing material, writing down any notes or the use of scratch paper is prohibited. You do get to make notes on a digital whiteboard within the exam software.

Stopping for any reason, including a restroom break, is prohibited.

With so many rules, you want to think carefully before taking the test at home. If you do plan to go this route, please visit Oracle's website for a complete set of rules and requirements.

THE DAY OF THE EXAM

When you go to take the exam, remember to bring two forms of

ID including one that is government issued. See Pearson's list of acceptable IDs here:

www.pearsonvue.com/policies/1S.pdf

Try not to bring too much extra with you as it will not be allowed into the exam room. While you will be allowed to check your belongings, it is better to leave extra items at home or in the car. You will not be allowed to bring paper, your phone, and the like into the exam room with you. Some centers are stricter than others. At one center, tissues were even taken away from us! Most centers allow keeping your ID and money. They watch you taking the exam, though, so don't even think about writing notes on money.

As we mentioned earlier, the exam center will give you writing materials to use during the exam, either scratch paper or a whiteboard. If you aren't given these materials, remember to ask. These items will be collected at the end of the exam.

FINDING OUT YOUR SCORE

In the past, you would find out right after finishing the exam if you passed. Now you have to wait nervously until you can check your score online. Many test takers check their score from a mobile device as they are walking out of the test center.

If you go onto the Pearson VUE website, it will just have a status of “Taken” rather than your result. Oracle uses a separate system for scores. You'll need to go to Oracle's CertView website to find out whether you passed and your score.

certview.oracle.com

It usually updates shortly after you finish your exam but can take up to an hour in some cases. In addition to your score, you'll also see objectives for which you got a question wrong. Once you have passed the 1Z0-816 exam
or the 1Z0-817 exam and fulfilled the required perquisites, the OCP 11 title will be granted within a few days.

Oracle has partnered with Acclaim, which is an Open Badge platform. Upon obtaining a certification from Oracle, you also receive a “badge” that you can choose to share publicly with current or prospective employers.

Objective Map

This book has been written to cover every objective on all four exams.

JAVA SE 11 PROGRAMMER I (1Z0‐815)

The following table provides a breakdown of this book's exam coverage for the Java SE 11 Programmer I (1Z0‐815) exam, showing you the chapter where each objective or subobjective is covered:

Exam Objective	**Chapter**
Understanding Java Technology and environment	
Describe Java Technology and the Java development	1
Identify key features of the Java language	1
Creating a Simple Java Program	
Create an executable Java program with a main class	1
Compile and run a Java program from the command line	1
Create and import packages	1
Working with Java Primitive Data Types and String APIs	
Declare and initialize variables (including casting and promoting primitive data types)	2, 3
Identify the scope of variables	2
Use local variable type inference	2
Create and manipulate Strings	5
Manipulate data using the StringBuilder class and its	5

methods	
Using Operators and Decision Constructs	
Use Java operators including the use of parentheses to override	3

Use Java operators including the use of parentheses to override operator precedence	3
Use Java control statements including if, if/else, switch	4
Create and use do/while, while, for and for each loops, including nested loops, use break and continue statements	4
Working with Java Arrays	
Declare, instantiate, initialize and use a one-dimensional array	5
Declare, instantiate, initialize and use a two-dimensional array	5
Describing and using Objects and Classes	
Declare and instantiate Java objects, and explain objects' lifecycles (including creation, dereferencing by reassignment, and garbage collection)	2
Define the structure of a Java class	1
Read or write to object fields	2
Creating and Using Methods	
Create methods and constructors with arguments and return values	7, 8
Create and invoke overloaded methods	7
Apply the static keyword to methods and fields	7
Applying Encapsulation	

Apply access modifiers	7
Apply encapsulation principles to a class	7
Reusing Implementations Through Inheritance	
Create and use subclasses and superclasses	8

Create and use subclasses and superclasses	8
Create and extend abstract classes	9
Enable polymorphism by overriding methods	8
Utilize polymorphism to cast and call methods, differentiating object type versus reference type	8
Distinguish overloading, overriding, and hiding	8
Programming Abstractly Through Interfaces	
Create and implement interfaces	9
Distinguish class inheritance from interface inheritance including abstract classes	9
Declare and use List and ArrayList instances	5, 6
Understanding Lambda Expressions	6
Handling Exceptions	
Describe the advantages of Exception handling and differentiate among checked, unchecked exceptions, and Errors	10
Create try-catch blocks and determine how exceptions alter program flow	10
Create and invoke a method that throws an exception	10
Understanding Modules	

Describe the Modular JDK	11
Declare modules and enable access between modules	11

	1
Describe how a modular project is compiled and run	11

JAVA SE 11 PROGRAMMER II (1Z0–816)

The following table provides a breakdown of this book's exam coverage for the Java SE 11 Programmer II (1Z0–816) exam, showing you the chapter where each objective or subobjective is covered.

Exam Objective	**Chapter**
Java Fundamentals	
Create and use final classes	12
Create and use inner, nested and anonymous classes	12
Create and use enumerations	12
Exception Handling and Assertions	
Use the try-with-resources construct	16

Create and use custom exception classes	16
Test invariants by using assertions	16
Java Interfaces	
Create and use interfaces with default methods	12
	2
Create and use interfaces with private methods	12
Generics and Collections	
Use wrapper classes, autoboxing and autounboxing	14
Create and use generic classes, methods with diamond notation and wildcards	14
Describe the Collections Framework and use key collection interfaces	14
Use Comparator and Comparable interfaces	14
Create and use convenience methods for collections	14
Functional Interfaces and Lambda Expressions	
Define and write functional interfaces	12
Create and use lambda expressions including statement lambdas, local-variable for lambda parameters	12
Java Stream API	

Describe the Stream interface and pipelines	1 5
Use lambda expressions and method references	1 5
Built‑in Functional Interfaces	
Use interfaces from the java.util.function package	1

Use interfaces from the java.util.function package	1 5
Use core functional interfaces including Predicate, Consumer, Function and Supplier	1 5
Use primitive and binary variations of base interfaces of java.util.function package	1 5
Lambda Operations on Streams	
Extract stream data using map, peek and flatMap methods	1 5
Search stream data using search findFirst, findAny, anyMatch, allMatch and noneMatch methods	1 5
Use the Optional class	1 5
Perform calculations using count, max, min, average and sum stream operations	1 5
Sort a collection using lambda expressions	1 5
Use Collectors with streams, including the groupingBy and partitioningBy operations	1 5
Migration to a Modular Application	

Migrate the application developed using a Java version prior to SE 9 to SE 11 including top-down and bottom-up migration, splitting a Java SE 8 application into modules for migration	17
Use jdeps to determine dependencies and identify ways to address the cyclic dependencies	17
Services in a Modular Application	
Describe the components of Services including directives	17
	7
Design a service type, load services using ServiceLoader, check for dependencies of the services including consumer and provider modules	17
Concurrency	
Create worker threads using Runnable, Callable and use an ExecutorService to concurrently execute tasks	18
Use java.util.concurrent collections and classes including CyclicBarrier and CopyOnWriteArrayList	18
Write thread-safe code	18
Identify threading problems such as deadlocks and livelocks	18
Parallel Streams	
Develop code that uses parallel streams	18
Implement decomposition and reduction with streams	18
I/O (Fundamentals and NIO.2)	

Read data from and write console and file data using I/O Streams	19
Use I/O Streams to read and write files	19
Read and write objects by using serialization	19
Use the Path interface to operate on file and directory paths	20

	0
Use the Files class to check, delete, copy or move a file or directory	20
Use the Stream API with Files	20
Secure Coding in Java SE Application	
Prevent Denial of Service in Java applications	22
Secure confidential information in Java application	22
Implement Data integrity guidelines- injections and inclusion and input validation	22
Prevent external attack of the code by limiting Accessibility and Extensibility, properly handling input validation, and mutability	22
Securely constructing sensitive objects	22
Secure Serialization and Deserialization	22
Database Applications with JDBC	

Connect to databases using JDBC URLs and DriverManager	21
Use PreparedStatement to perform CRUD operations	21
Use PreparedStatement and CallableStatement APIs to perform database operations	21
Localization	
Use the Locale class	1

Use the Locale class	16
Use resource bundles	16
Format messages, dates, and numbers with Java	16
Annotations	
Describe the purpose of annotations and typical usage patterns	13
Apply annotations to classes and methods	13
Describe commonly used annotations in the JDK	13
Declare custom annotations	13

UPGRADE OCP JAVA 6, 7 & 8 TO JAVA SE 11 DEVELOPER (1Z0–817)

The following table provides a breakdown of this book's exam coverage for the Upgrade OCP Java 6, 7 & 8 to Java SE 11 Developer (1Z0–817) exam, showing you the chapter where each objective or subobjective is covered.

Exam Objective	**Chapter**
Understanding Modules	
Describe the Modular JDK	11
Declare modules and enable access between modules	11
Describe how a modular project is compiled and run	11
Migration to a Modular Application	
Migrate the application developed using a Java version prior to SE 9 to SE 11 including top-down and bottom-up migration, splitting a Java SE 8 application into modules for migration	17
Use jdeps to determine dependencies and identify way to address the cyclic dependencies	17
Services in a Modular Application	
Describe the components of Services including directives	17
Design a service type, load services using ServiceLoader, check for dependencies of the services including consumer module and provider modules	17
Local Variable Type Inference	

Use local variable type inference	2
Create and use lambda expressions with local variable type inferred parameters	2
Java Interfaces	
Create and use methods in interfaces	12
Define and write functional interfaces	12
Lambda Expressions	

Create and use lambda expressions	12
Use lambda expressions and method references	15
Use built-in functional interfaces including Predicate, Consumer, Function, and Supplier	15
Use primitive and binary variations of base interfaces of java.util.function package	15
Lambda Operations on Streams	
Extract stream data using map, peek and flatMap methods	15
Search stream data using search findFirst, findAny, anyMatch, allMatch and noneMatch methods	15
Use the Optional class	15
Perform calculations using count, max, min, average and sum stream operations	15

Sort a collection using lambda expressions	15
Use Collectors with streams, including the groupingBy and partitioningBy operation	15
Parallel Streams	
Develop the code that use parallel streams	18
Implement decomposition and reduction with streams	18

Java File IO (NIO.2)	
Use Path interface to operate on file and directory paths	20
Use Files class to check, delete, copy or move a file or directory	20
Use Stream API with Files	20
Language Enhancements	
Use try-with-resources construct	16
Develop code that handles multiple Exception types in a single catch block	16

JAVA FOUNDATIONS (1Z0-811)

The following table provides a breakdown of this book's exam coverage for the Java Foundations (1Z0-811) exam, showing you the chapter where each objective or subobjective is covered.

A few topics are on the Java Foundations exam, but not the 1Z0-815. Those are covered here:

www.selikoff.net/java-foundations

Additionally, the objectives may be updated when Oracle updates the Java Foundations exam for Java 11. Check our website for those updates as well.

Exam Objective	**Chap te r**
What is Java?	

What is Java?	
Describe the features of Java	1
Describe the real-world applications of Java	1 + online
Java Basics	
Describe the Java Development Kit (JDK) and the Java Runtime Environment (JRE)	1
Describe the components of object-oriented programming	1
Describe the components of a basic Java program	1
Compile and execute a Java program	1
Basic Java Elements	
Identify the conventions to be followed in a Java program	1
Use Java reserved words	2
Use single-line and multi-line comments in java programs	2

Import other Java packages to make them accessible in your code	1
Describe the java.lang package	1
Working with Java Data Types	
Declare and initialize variables including a variable using final	2
Cast a value from one data type to another including automatic and manual promotion	2
Declare and initialize a String variable	2
Working with Java Operators	
Use basic arithmetic operators to manipulate data including +, -, *, /, and %	2
Use the increment and decrement operators	2
Use relational operators including ==, !=, >, >=, <, and <=	2
Use arithmetic assignment operators	2
Use conditional operators including &&, ||, and ?	2
Describe the operator precedence and use of parentheses	2
Working with the String Class	
Develop code that uses methods from the String class	5
Format Strings using escape sequences including %d, %n, and %s	Online
Working with Random and Math Classes	
Use the Random class	Online

Use the Math class	5
Using Decision Statements	
Use the decision making statement (if-then and if-then-else)	4
Use the switch statement	4
Compare how == differs between primitives and objects	3
Compare two String objects by using the compareTo and equals methods	5
Using Looping Statements	
Describe looping statements	4

Describe looping statements	4
Use a for loop including an enhanced for loop	4
Use a while loop	4
Use a do- while loop	4
Compare and contrast the for, while, and do-while loops	4
Develop code that uses break and continue statements	4
Debugging and Exception Handling	
Identify syntax and logic errors	1, 2, 3, 4, 5
Use exception handling	10
Handle common exceptions thrown	10
Use try and catch blocks	10
Arrays and ArrayLists	
Use a one-dimensional array	5

Create and manipulate an ArrayList	5
Traverse the elements of an ArrayList by using iterators and loops including the enhanced for loop	5 + online
Compare an array and an ArrayList	5
Classes and Constructors	
Create a new class including a main method	1
Use the private modifier	7
Describe the relationship between an object and its members	8

Describe the relationship between an object and its members	8
Describe the difference between a class variable, an instance variable, and a local variable	2, 8
Develop code that creates an object's default constructor and modifies the object's fields	8
Use constructors with and without parameters	8
Develop code that overloads constructors	8
Java Methods	
Describe and create a method	7
Create and use accessor and mutator methods	7
Create overloaded methods	7
Describe a static method and demonstrate its use within a program	7

Assessment Tests

Use the following assessment tests to gauge your current level of skill in Java for 1Z0-815 and 1Z0-816. These tests are designed to highlight some topics for your strengths and weaknesses so that you know which chapters you might want to read multiple times. Even if you do well on the assessment tests, you should still read the book from cover to cover, as the real exams are quite challenging.

If you are taking the 1Z0-817 exam, you can still take the 1Z0-815 and 1Z0-816 assessments respectively. If you get a question wrong on the Part I assessment test, you should review the associated chapter to make sure you understand the material. Remember, the 1Z0-817 exam is cumulative. On the other hand, if you get a question wrong on the Part II assessment test, you should check the list of objectives and see if it is in scope for the upgrade exam. For example, the 1Z0-817 exam will not test you on annotations or security.

PART I: EXAM 1Z0-815

1. What is the result of the following program?

```
        1: public class MathFunctions {
2: public static void addToInt(int x, int amountToAdd) {
3: x = x + amountToAdd;
4: }
                    5: public static void main(String[] args) {
6: var a = 15;
7: var b = 10;
8: MathFunctions.addToInt(a, b);
9: System.out.println(a); } }
```

1. 10
2. 15
3. 25
4. Compiler error on line 3
5. Compiler error on line 8
6. None of the above

2. What is the output of the following program? (Choose all that apply.)

```
                  1: interface HasTail { int getTailLength(); }
             2: abstract class Puma implements HasTail {
3: protected int getTailLength() { return 4; } 4: }
             5: public class Cougar implements HasTail {
                   6: public static void main(String[] args) {
7: var puma = new Puma();
8: System.out.println(puma.getTailLength()); 9: }
10: public int getTailLength(int length) { return 2; } 11: }
```

1. 2
2. 4
3. The code will not compile because of line 3.
4. The code will not compile because of line 5.
5. The code will not compile because of line 7.
6. The code will not compile because of line 10.
7. The output cannot be determined from the code provided. 3. What is the output of the following code snippet?

```
int moon = 9, star = 2 + 2 * 3;
float sun = star>10 ? 1 : 3;
double jupiter = (sun + moon) - 1.0f;
int mars = --moon <= 8 ? 2 : 3;
System.out.println(sun+"-"+jupiter+"-"+mars);
```

1. 1-11-2
2. 3.0-11.0-2
3. 1.0-11.0-3
4. 3.0-13.0-3
5. 3.0f-12-2
6. The code does not compile because one of assignments requires an explicit numeric cast.

4. How many times is the word true printed?

```
var s1 = "Java";
var s2 = "Java";
var s3 = "Ja".concat("va");
var s4 = s3.intern();
var sb1 = new StringBuilder();
sb1.append("Ja").append("va");

System.out.println(s1 == s2);
System.out.println(s1.equals(s2));
System.out.println(s1 == s3);
System.out.println(s1 == s4);
System.out.println(sb1.toString() == s1);
System.out.println(sb1.toString().equals(s1));
```
1. Once
2. Twice
3. Three times

4. Four times

5. Five times

6. Six times

7. The code does not compile.

5. The following code appears in a file named `Flight.java`. What is the result of compiling this source file?

```
1: public class Flight {
2: private FlightNumber number;
3:
          4: public Flight(FlightNumber number) {
5: this.number = number;
6: } }
7: public class FlightNumber {
8: public int value;
9: public String code; }
```

1. The code compiles successfully and two bytecode files are generated: `Flight.class` and `FlightNumber.class`.

2. The code compiles successfully and one bytecode file is generated: `Flight.class`.

3. A compiler error occurs on line 2.

4. A compiler error occurs on line 4.

5. A compiler error occurs on line 7.

6. Which of the following will run a modular program?

1. `java -cp modules mod/class`

2. `java -cp modules -m mod/class`

3. `java -cp modules -p mod/class`

4. `java -m modules mod/class`

5. `java -m modules -p mod/class`

6. `java -p modules mod/class`

7. `java -p modules -m mod/class`

7. What is the result of executing the following code snippet?

```
         final int score1 = 8, score2 = 3;
char myScore = 7;
switch (myScore) {
     default:
     score1:
                    2: 6: System.out.print("great-");
                    4: System.out.print("good-"); break;
score2:
        1: System.out.print("not good-");
}
```

1. great-good-
2. good-
3. not good-
4. great-good-not-good-
5. The code does not compile because default is not a keyword in Java. 6. The code does not compile for a different reason.

8. Which of the following lines can fill in the blank to print true? (Choose all that apply.)

```
10: public static void main(String[] args) {
11: System.out.println(____________________________); 12: }
13: private static boolean test(Predicate<Integer> p) { 14: return p.test(5);
15: }
```

1. test(i -> i == 5)
2. test(i -> {i == 5;})
3. test((i) -> i == 5)
4. test((int i) -> i == 5)
5. test((int i) -> {return i == 5;})
6. test((i) -> {return i == 5;})

9. Which of the following are valid instance members of a class? (Choose all that apply.)

1. var var = 3;
2. Var case = new Var();
3. void var() {}
4. int Var() { var _ = 7; return _;}
5. String new = "var";
6. var var() { return null; }

10. Which of the following types can be inserted into the blank that allows the program to compile successfully? (Choose all that apply.)

```
1: import java.util.*;
2: interface CanSwim {}
        3: class Amphibian implements CanSwim {}
                4: abstract class Tadpole extends Amphibian {}
5: public class FindAllTadPole {
```

```
6: public static void main(String[] args) {
7: var tadpoles = new ArrayList<Tadpole>(); 8: for (Amphibian
amphibian : tadpoles) {
9: ____________ tadpole = amphibian;
10: } } }
```

1. CanSwim
2. Boolean
3. Amphibian
4. Tadpole
5. Object
6. None of the above; the program contains a compilation error.

11. Which of the following expressions compile without error? (Choose all that apply.)

1. int monday = 3 + 2.0;
2. double tuesday = 5_6L;
3. boolean wednesday = 1 > 2 ? !true;
4. short thursday = (short)Integer.MAX_VALUE;
5. long friday = 8.0L;
6. var saturday = 2_.0;
7. None of the above

12. Suppose you have a module named com.vet. Where could you place the following module-info.java file to create a valid module?

```
public module com.vet {
    exports com.vet;
}
```

1. At the same level as the com folder
2. At the same level as the vet folder
3. Inside the vet folder
4. None of the above

13. What is the result of compiling and executing the following program?

```
1: public class FeedingSchedule {
2: public static void main(String[] args) {
3: var x = 5;
4: var j = 0;
5: OUTER: for (var i = 0; i < 3;)
6: INNER: do {
7: i++;
8: x++;
9: if (x> 10) break INNER;
10: x += 4;
11: j++;
12: } while (j <= 2);
13: System.out.println(x);
14: } }
```

1. 10
2. 11
3. 12
4. 17
5. The code will not compile because of line 5.
6. The code will not compile because of line 6.

14. Which statement about the following method is true?

```
                        5: public static void main(String... unused) {
6: System.out.print("a");
7: try (StringBuilder reader = new StringBuilder()) { 8: System.out.print("b");
                          9: throw new IllegalArgumentException();
10: } catch (Exception e || RuntimeException e) { 11:
System.out.print("c");
                       12: throw new FileNotFoundException();
13: } finally {
14: System.out.print("d");
15: } }
```

1. It compiles and prints `abc`.
2. It compiles and prints `abd`.
3. It compiles and prints `abcd`.
4. One line contains a compiler error.
5. Two lines contain a compiler error.
6. Three lines contain a compiler error.
7. It compiles but prints an exception at runtime.

15. Which of the following are true statements? (Choose all that
apply.) 1. The JDK contains a compiler.
2. The JVM contains a compiler.
3. The `javac` command creates a file containing bytecode.
4. The `java` command creates a file containing bytecode.
5. The JDK is contained in the JVM.
6. The JVM is contained in the JDK.

16. Which lines in Tadpole give a compiler error? (Choose all that apply.)

```
1: package animal;
2: public class Frog {
3: protected void ribbit() { }
4: void jump() { }
5: }

1: package other;
2: import animal.*;
3: public class Tadpole extends Frog {
                              4: public static void main(String[] args) {
5: Tadpole t = new Tadpole();
6: t.ribbit();
7: t.jump();
8: Frog f = new Tadpole();
9: f.ribbit();
10: f.jump();
11: } }
```

1. 5
2. 6
3. 7
4. 8
5. 9
6. 10

17. What is the output of the following program?

```
1: class Deer {
                              2: public Deer() {System.out.print("Deer");}
3: public Deer(int age) {System.out.print("DeerAge");} 4: protected boolean
hasHorns() { return false; } 5: }
6: public class Reindeer extends Deer {
7: public Reindeer(int age)
{System.out.print("Reindeer");}
8: public boolean hasHorns() { return true; } 9: public static void
main(String[] args) { 10: Deer deer = new Reindeer(5);
11: System.out.println("," + deer.hasHorns()); 12: } }
```

1. ReindeerDeer,false
2. DeerAgeReindeer,true
3. DeerReindeer,true
4. DeerReindeer,false
5. ReindeerDeer,true
6. DeerAgeReindeer,false

7. The code will not compile because of line 4.
8. The code will not compile because of line 12.

18. What is printed by the following code? (Choose all that apply.)

```
int[] array = {6,9,8};
List<Integer> list = new ArrayList<>();
list.add(array[0]);
list.add(array[2]);
list.set(1, array[1]);
list.remove(0);
System.out.println(list);
System.out.println("C" + Arrays.compare(array, new int[] {6, 9,
      8}));
System.out.println("M" + Arrays.mismatch(array, new int[] {6, 9,
      8}));
```

1. [8]
2. [9]
3. [Ljava.lang.String;@160bc7c0
4. C-1
5. C0
6. M-1
7. M0