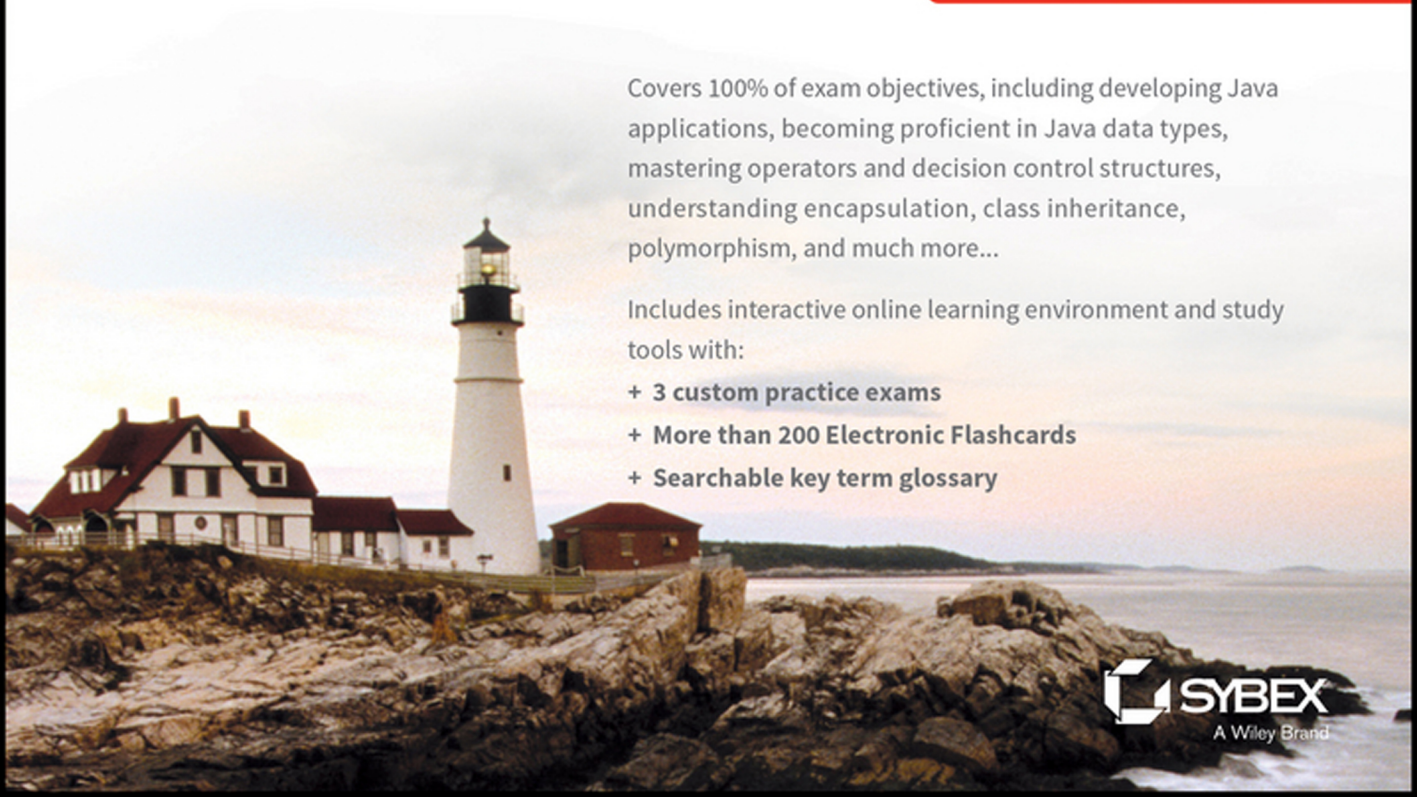
Jeanne Boyarsky and Scott Selikoff

# OCA

## Oracle Certified Associate Java® SE 8 Programmer I

# STUDY GUIDE

**EXAM 1Z0-808**



Covers 100% of exam objectives, including developing Java applications, becoming proficient in Java data types, mastering operators and decision control structures, understanding encapsulation, class inheritance, polymorphism, and much more...

Includes interactive online learning environment and study tools with:

+ **3 custom practice exams**
+ **More than 200 Electronic Flashcards**
+ **Searchable key term glossary**

SYBEX
A Wiley Brand

# OCA: Oracle®

# Certified Associate Java® SE 8 Programmer I

## Study Guide

## Exam 1Z0-808

# OCA: Oracle®

# Certified Associate Java® SE 8 Programmer I

## Study Guide

## Exam 1Z0-808

Jeanne Boyarsky

Scott Selikoff

Senior Acquisitions Editor: Kenyon Brown
Development Editor: Alexa Murphy
Technical Editors: Ernest Friedman-Hill, Matt Dalen
Production Editor: Rebecca Anderson
Copy Editor: Liz Welch
Editorial Manager: Pete Gaughan
Vice President and Executive Group Publisher: Richard Swadley
Associate Publisher: Jim Minatel
Production Manager: Kathleen Wisor
Media Supervising Producer: Rich Graves
Book Designers: Judy Fung and Bill Gibson
Proofreader: Scott Klemp, Word One New York
Indexer: Ted Laux
Project Coordinator, Cover: Patrick Redmond
Cover Designer: Wiley
Cover Image: ©Getty Images Inc./Jeremy Woodhouse

Published simultaneously in Canada

ISBN: 978-1-118-95740-0
ISBN: 978-1-118-95741-7 (ebk.)
ISBN: 978-1-118-95742-4 (ebk.)



**Library of Congress Control Number:** 2014954685



10 9 8 7 6 5 4 3 2 1

Dear Reader,

Thank you for choosing *OCA: Oracle Certified Associate Java SE 8 Programmer I Study Guide*. This book is part of a family of premium-quality Sybex books, all of which are written by outstanding authors who combine practical experience with a gift for teaching.

Sybex was founded in 1976. More than 30 years later, we're still committed to producing consistently exceptional books. With each of our titles, we're working hard to set a new standard for the industry. From the paper we print on, to the authors we work with, our goal is to bring you the best books available.

I hope you see all that reflected in these pages. I'd be very interested to hear your comments and get your feedback on how we're doing. Feel free to let me know what you think about this or any other Sybex book by sending me an email at `contactus@wiley.com`. If you think you've found a technical error in this book, please visit `http://sybex.custhelp.com`. Customer feedback is critical to our efforts at Sybex.

Best regards,

Chris Webb
Associate Publisher
Sybex, an Imprint of Wiley

*To the programmers on FIRST robotics team 694.*

*—Jeanne*

*To my wife and the two little bundles of joy she is carrying.*

*—Scott*

# Acknowledgments

Jeanne and Scott would like to thank numerous individuals for their contribution to this book. Thank you to Developmental Editor Alexa Murphy for teaching us about Wiley's publishing process and making the book better in so many ways. Thank you to Ernest Friedman-Hill for being our Technical Editor as we wrote our first book. Ernest pointed out many subtle errors in addition to the big ones. And thank you to Matt Dalen for being our Technical Proofer and finding the errors we managed to sneak by Ernest. This book also wouldn't be possible without many people at Wiley, including Jeff Kellum, Kenyon Brown, Pete Gaughan, Rebecca Anderson, and so many others.

Jeanne would personally like to thank Chris Kreussling for knowing almost a decade ago that she would someday write a book. Erik Kariyev motivated her to write her first table of contents ever. Countless CodeRanch.com moderators warned Jeanne about how much work writing a book is to get her to the point where she was ready. Michael Ernest gave her extra advice on the Wiley process. Bert Bates let Jeanne dip her toe in by contributing to his Java 7 book and she learned a ton in the process. Scott was a great co-author and was available to bounce ideas off of or remind her to follow her own advice. Finally, Jeanne would like to thank all of the new programmers at CodeRanch.com and FIRST robotics team 694 for the constant reminders of how new programmers think.

Scott could not have reached this point without the help of a small army of people, led by his perpetually understanding wife Patti, without whose love and support this book would never have been possible. Professor Johannes Gehrke of Cornell University always believed in him and knew he would excel in his career. Jeanne's patience and guidance as co-author was invaluable while Scott adjusted to the learning curve of writing a book. Matt Dalen has been a wonderful friend and sounding board over the last year. Joel McNary introduced him to CodeRanch.com and encouraged him to post regularly, a step that changed his life. 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.

# About the Authors

**Jeanne Boyarsky** has worked as a Java developer for over 12 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 codebase. Jeanne also mentors the programming division of a FIRST robotics team, where she works with students just getting started with Java.

Jeanne got her Bachelor of Arts in 2002 and her Master's in Computer Information Technology 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.coderanch.com/how-to/java/BioJeanneBoyarsky`.

**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 database-driven systems, web-based applications, and service-oriented architectures.

A native of Toms River, NJ, Scott achieved his Bachelor of Arts from Cornell University in Mathematics and Computer Science in 2002, after 3 years of study. In 2003, he received his Master's of Engineering 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 The Server Side Java Symposium. Scott lives in New Jersey with his loving wife and two very playful dogs, a Siberian husky named Webby and standard poodle named Georgette. You can find out more about Scott at `www.linkedin.com/in/selikoff`.

Jeanne and Scott are both moderators on the CodeRanch.com forums and can be reached there for questions and comments. They also co-author a technical blog called Down Home Country Coding at `www.selikoff.net`.

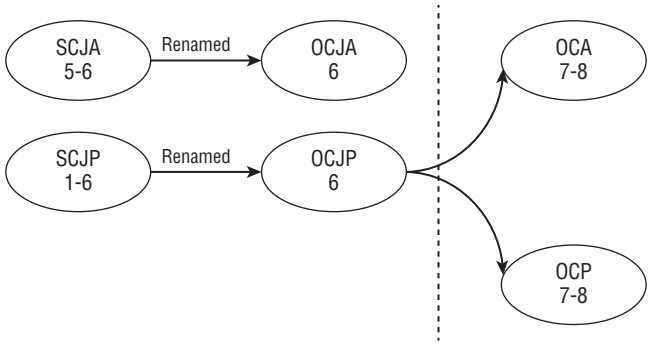
# Contents at a Glance

# Contents

# Introduction

Java, "born" in 1995, is now just about 20 years old. As with anything 20 years old, there is a good amount of history and variation between 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 8 Oracle Certified Associate (OCA) exam.

If you read about "the exam" on the Web, you may see information about the older names for the exam. We've showed the changes in name. Here's what happened. Sun Microsystems used to have two exams. The SCJP (Sun Certified Java Programmer) was meant for programmers and the SCJA (Sun Certified Java Associate) was meant for those who wanted broader knowledge. When Oracle bought Sun Microsystems, they changed all the names from Sun to Oracle, giving us the OCJP and OCJA.




Then Oracle made two strategic decisions with Java 7. They decided to stop updating the OCJA exam. They also decided to cover more on in the programmer space and split it into two exams. Now you first take the OCAJP (Oracle Certified Associate Java Programmer), also known as Java Programmer I, or OCA. That's what this book is about. Then you take the OCPJP (Oracle Certified Professional Java Programmer), also known as Java Programmer II, or OCP. There's also an upgrade exam in case you took an older version of the SCJP or OCPJP and want to upgrade. Most people refer to the current exams as OCA 8, OCP 8, and the Java 8 upgrade exam. We mention when a topic is split between the OCA and OCP so you know which parts are more advanced.

We try to keep the history to a minimum in this book. There are some places on the exam where you need to know both an "old way" and a "new way" of doing things. When that happens, we will be sure to tell you what version of Java introduced it. We will also let you know about topics that are not on the exam anymore in case you see questions in the older free online mock exams.

# The OCA Exam

All you need to do to earn the Oracle Certified Associate Java SE 8 Programmer certification is to pass the exam! That's it.

Oracle has a tendency to fiddle with the length of the exam and the passing score once it comes out. Since it's pretty much a guarantee that whatever we tell you here will become obsolete, we will give you a feel for the range of variation. The OCA exam has varied between 60 and 90 questions since it was introduced. The score to pass has varied between 60 percent and 80 percent. The time allowed to take the exam has varied from two hours to two-and-a-half hours.

Oracle has a tendency to "tweak" the exam objectives over time as well. They do make minor additions and removals from what is covered on the exam. Although this tends to affect the OCP exam more than the OCA exam, there are a few topics that were added to the OCA for Java 8. It wouldn't be a surprise for Oracle to make changes.

Although there will likely be minor changes to the scope of the exam, it certainly isn't a secret. We've created a book page on our blog: `www.selikoff.net/oca`. If there are any changes to the topics on the exam after this book is published, we will note them there.

That book page also contains a link to the official exam page so that you can check the length and passing score that Oracle has chosen for the moment.

# 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`. Choose IT and then Oracle. 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 someone complaining about their broken laptop nearby!

At this time, you can reschedule the exam without penalty until up to 24 hours before. This means that 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 on the Pearson VUE website. This may change, so check the rules before paying.

# 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 what is acceptable ID at `http://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 so forth 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.

The exam center will give you writing materials to use during the exam. These are used as scratch paper during the exam to figure out answers and keep track of your thought process. The exam center will dispose of them at the end. Notice how we said "writing materials" rather than "pen and paper." Some centers still give pen and paper. Most give a small erasable board and a dry erase marker. If you have a preference to which you receive, call the testing center in advance to inquire.

# 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.

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 `http://certview.oracle.com` to find out whether you passed and your score. It doesn't update immediately upon taking the test, but we haven't heard of it taking more than an hour. In addition to your score, you'll also see objectives for which you got a question wrong and instructions on how to get a hardcopy certificate.

At some point, you'll get an electronic certificate and some more time after that you'll receive your printed certificate. Sound vague? It is. The times reported to receive certificates vary widely.

# Exam Questions

The OCA exam consists of multiple-choice questions. There are typically five or six 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.

Note that exam questions will sometimes have line numbers that begin with numbers higher than 1. This is to indicate that you are looking at a code snippet rather than a complete class. We follow this convention as well to get you used to it.

If you read about older versions of the exam 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. There was also a bug in the exam software that caused your answers to get lost if you reviewed them again. Luckily, these are no longer on the exam.

# Getting Started

We recommend reading Appendix B, "Study Tips," before diving into the technical material in this book. Knowing how to approach studying will help you make better use of your study time.

Next, make sure you have downloaded version 8 of the JDK. If you learned Java some time ago, you might have version 7 or even earlier. There have been both big and small changes to the language. You could get a question wrong if you study with the wrong version.

Also, please check our book page to make sure Oracle hasn't changed the objectives. For example, if Oracle decided that lambdas weren't on the exam, you'd want to know that before studying. We will post any updates that you should know about at `www.selikoff.net/oca`.

# Getting Help

Both of the authors are moderators at CodeRanch.com. CodeRanch.com is a very large and active programming forum that is very friendly toward Java beginners. It has a forum just for this exam called OCAJP. 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.

# Who Should Buy This Book

If you want to become certified as a Java programmer, 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, 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 consists of six chapters, plus supplementary information: a glossary, this introduction, three appendices, and the assessment test after the introduction. You might have noticed that there are more than six exam objectives. We split up 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.

The chapters are organized as follows:

- Chapter 1, "Java Building Blocks," covers the basics of Java such as scoping variables and how to run a program. It also includes calling methods and types of variables.
- Chapter 2, "Operators and Statements," focuses on the core logical constructs such as conditionals and loops. It also talks about the meaning and precedence of operators.
- Chapter 3, "Core Java APIs," introduces you to array, `ArrayList`, `String`, `StringBuilder`, and various date classes.
- Chapter 4, "Methods and Encapsulation," explains how to write methods, including access modifiers. It also shows how to call lambdas.
- Chapter 5, "Class Design," adds interfaces and superclasses. It also includes casting and polymorphism.
- Chapter 6, "Exceptions," shows the different types of exception classes and how to use them.

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 readily familiar with the key points of each chapter and be able to explain them in detail.

**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 Appendix A.

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.


The review questions, assessment test, 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 learn the underlying 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, even for years.

# Free Online Learning Environment

This book provides a free online interactive learning environment and test bank with several additional elements. The online test bank includes:

**Sample Tests** All of the questions in this book, including the 20-question assessment test at the end of this introduction and over 130 questions that make up the Review Question sections for each chapter. In addition, there are three 60-question Practice Exams to test your knowledge of the material. The online test bank runs on multiple devices.

**Electronic Flashcards** Over 200 questions in flashcard format (a question followed by a single correct answer). You can use these to reinforce your learning and provide last-minute test prep before the exam.

**Glossary** The key terms from this book and their definitions are available as a fully searchable PDF.


Go to `www.sybex.com/go/ocajavase8` to register and gain access to this comprehensive study tool package.

# Conventions Used in This Book

This book uses certain typographic styles in order to help you quickly identify important information and to avoid confusion about the meaning of words, such as onscreen 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 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**

**Real World Scenario**

A real world scenario describes a task or an example that's particularly grounded in the real world. Although interesting, the scenario will not show up on the exam.

# OCA Exam Objectives

*OCA: Oracle Certified Associate Java SE 8 Programmer I Study Guide: Exam 1Z0-808* has been written to cover every OCA exam objective. The following table provides a breakdown of this book's exam coverage, showing you the chapter where each objective or sub-objective is covered:

| Exam Objective | Chapter |
|---|---|
| ■ **Java Basics** | |
| Define the scope of variables | 1 |
| Define the structure of a Java class | 1 |
| Create executable Java applications with a main method; run a Java program from the command line, including console output | 1 |
| Import other Java packages to make them accessible in your code | 1 |
| Compare and contrast the features and components of Java such as platform independence, object orientation, encapsulation, etc. | 1 |
| ■ **Working with Java Data Types** | |
| Declare and initialize variables (including casting of primitive data types) | 1 |
| Differentiate between object reference variables and primitive variables | 1 |
| Know how to read or write to object fields | 1 |
| Explain an Object's Lifecycle (creation, "dereference by reassignment," and garbage collection) | 1 |
| Develop code that uses wrapper classes such as Boolean, Double, and Integer | 1 |
| ■ **Using Operators and Decision Constructs** | |
| Use Java operators, including parentheses to override operator precedence | 2 |
| Test equality between Strings and other objects using == and equals () | 3 |
| Create if and if/else and ternary constructs | 2 |
| Use a switch statement | 2 |
| ■ **Creating and Using Arrays** | |
| Declare, instantiate, initialize, and use a one-dimensional array | 3 |
| Declare, instantiate, initialize, and use multi-dimensional array | 3 |

| Exam Objective | Chapter |
|---|---|
| ■ **Using Loop Constructs** | |
| Create and use while loops | 2 |
| Create and use for loops including the enhanced for loop | 2 |
| Create and use do/while loops | 2 |
| Compare loop constructs | 2 |
| Use break and continue | 2 |
| ■ **Working with Methods and Encapsulation** | |
| Create methods with arguments and return values, including overloaded methods | 4 |
| Apply the static keyword to methods and fields | 4 |
| Create and overload constructors, including impact on default constructors | 4 |
| Apply access modifiers | 4 |
| Apply encapsulation principles to a class | 4 |
| Determine the effect upon object references and primitive values when they are passed into methods that change the values | 4 |
| ■ **Working with Inheritance** | |
| Describe inheritance and its benefits | 5 |
| Develop code that demonstrates the use of polymorphism, including overriding and object type versus reference type | 5 |
| Determine when casting is necessary | 5 |
| Use super and this to access objects and constructors | 5 |
| Use abstract classes and interfaces | 5 |
| ■ **Handling Exceptions** | |
| Differentiate among checked exceptions, unchecked exceptions, and Errors | 6 |
| Create a try-catch block and determine how exceptions alter normal program flow | 6 |

*(continued)*

| Exam Objective | Chapter |
|---|---|
| Describe the advantages of Exception handling | 6 |
| Create and invoke a method that throws an exception | 6 |
| Recognize common exception classes (such as NullPointerException, ArithmeticException, ArrayIndexOutOfBoundsException, ClassCastException) | 6 |
| ■ **Working with Selected Classes from the Java API** | |
| Manipulate data using the StringBuilder class and its methods | 3 |
| Creating and manipulating Strings | 3 |
| Create and manipulate calendar data using classes from java.time.LocalDateTime, java.time.LocalDate, java.time.LocalTime, java.time.format.DateTimeFormatter, java.time.Period | 3 |
| Declare and use an ArrayList of a given type | 3 |
| Write a simple Lambda expression that consumes a Lambda Predicate expression | 4 |

# Assessment Test

1. What is the result of the following class? (Choose all that apply)

```
1: public class _C {
2:  private static int $;
3:  public static void main(String[] main) {
4:    String a_b;
5:    System.out.print($);
6:    System.out.print(a_b);
7:  } }
```

   **A.** Compiler error on line 1.
   **B.** Compiler error on line 2.
   **C.** Compiler error on line 4.
   **D.** Compiler error on line 5.
   **E.** Compiler error on line 6.
   **F.** `0null`
   **G.** `nullnull`

2. What is the result of the following code?

```
String s1 = "Java";
String s2 = "Java";
StringBuilder sb1 = new StringBuilder();
sb1.append("Ja").append("va");
System.out.println(s1 == s2);
System.out.println(s1.equals(s2));
System.out.println(sb1.toString() == s1);
System.out.println(sb1.toString().equals(s1));
```

   **A.** `true` is printed out exactly once.
   **B.** `true` is printed out exactly twice.
   **C.** `true` is printed out exactly three times.
   **D.** `true` is printed out exactly four times.
   **E.** The code does not compile.

3. What is the output of the following code? (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 extends Puma {
```

```
6:    public static void main(String[] args) {
7:      Puma puma = new Puma();
8:      System.out.println(puma.getTailLength());
9:    }
10:
11:  public int getTailLength(int length) {return 2;}
12: }
```

**A.** 2

**B.** 4

**C.** The code will not compile because of line 3.

**D.** The code will not compile because of line 5.

**E.** The code will not compile because of line 7.

**F.** The code will not compile because of line 11.

**G.** The output cannot be determined from the code provided.

**4.** What is the output of the following program?

```
1: public class FeedingSchedule {
2:  public static void main(String[] args) {
3:    boolean keepGoing = true;
4:    int count = 0;
5:    int x = 3;
6:    while(count++ < 3) {
7:      int y = (1 + 2 * count) % 3;
8:      switch(y) {
9:         default:
10:         case 0: x -= 1; break;
11:         case 1: x += 5;
12:      }
13:    }
14:  System.out.println(x);
15: } }
```

**A.** 4

**B.** 5

**C.** 6

**D.** 7

**E.** 13

**F.** The code will not compile because of line 7.

**5.** What is the output of the following code snippet?

```
13: System.out.print("a");
14: try {
15:   System.out.print("b");
16:   throw new IllegalArgumentException();
17: } catch (RuntimeException e) {
18:   System.out.print("c");
19: } finally {
20:   System.out.print("d");
21: }
22: System.out.print("e");
```

**A.** abe

**B.** abce

**C.** abde

**D.** abcde

**E.** The code does not compile.

**F.** An uncaught exception is thrown.

**6.** 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:     int a = 15;
7:     int b = 10;
8:     MathFunctions.addToInt(a, b);
9:     System.out.println(a);   } }
```

**A.** 10

**B.** 15

**C.** 25

**D.** Compiler error on line 3.

**E.** Compiler error on line 8.

**F.** None of the above.

**7.** What is the result of the following code?

```
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);
```

**A.** `[8]`

**B.** `[9]`

**C.** Something like `[Ljava.lang.String;@160bc7c0`

**D.** An exception is thrown.

**E.** The code does not compile.

**8.** What is the output of the following code?

```
1: public class Deer {
2:  public Deer() { System.out.print("Deer"); }
3:  public Deer(int age) { System.out.print("DeerAge"); }
4:  private boolean hasHorns() { return false; }
5:  public static void main(String[] args) {
6:    Deer deer = new Reindeer(5);
7:    System.out.println(","+deer.hasHorns());
8:  }
9: }
10: class Reindeer extends Deer {
11:  public Reindeer(int age) { System.out.print("Reindeer"); }
12:  public boolean hasHorns() { return true; }
13: }
```

**A.** `DeerReindeer,false`

**B.** `DeerReindeer,true`

**C.** `ReindeerDeer,false`

**D.** `ReindeerDeer,true`

**E.** `DeerAgeReindeer,false`

**F.** `DeerAgeReindeer,true`

**G.** The code will not compile because of line 7.

**H.** The code will not compile because of line 12.

**9.** Which of the following statements are true? (Choose all that apply)

**A.** Checked exceptions are intended to be thrown by the JVM (and not the programmer).

**B.** Checked exceptions are required to be caught or declared.

**C.** Errors are intended to be thrown by the JVM (and not the programmer).

**D.** Errors are required to be caught or declared.

**E.** Runtime exceptions are intended to be thrown by the JVM (and not the programmer).

**F.** Runtime exceptions are required to be caught or declared.

**10.** Which are true of the following code? (Choose all that apply)

```
1:  import java.util.*;
2:  public class Grasshopper {
3:  public Grasshopper(String n) {
4:     name = n;
5:  }
6:  public static void main(String[] args) {
7:    Grasshopper one = new Grasshopper("g1");
8:    Grasshopper two = new Grasshopper("g2");
9:    one = two;
10:   two = null;
11:   one = null;
12: }
13:   private String name; }
```

**A.** Immediately after line 9, no grasshopper objects are eligible for garbage collection.

**B.** Immediately after line 10, no grasshopper objects are eligible for garbage collection.

**C.** Immediately after line 9, only one grasshopper object is eligible for garbage collection.

**D.** Immediately after line 10, only one grasshopper object is eligible for garbage collection.

**E.** Immediately after line 11, only one grasshopper object is eligible for garbage collection.

**F.** The code compiles.

**G.** The code does not compile.

**11.** What is the output of the following program?

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

**A.** 10

**B.** 12

**C.** 13

**D.** 17

**E.** The code will not compile because of line 4.

**F.** The code will not compile because of line 6.

**12.** What is the result of the following program?

```
1: public class Egret {
2:   private String color;
3:   public Egret() {
4:     this("white");
5:   }
6:   public Egret(String color) {
7:     color = color;
8:   }
9:   public static void main(String[] args) {
10:    Egret e = new Egret();
11:    System.out.println("Color:" + e.color);
12:   }
13: }
```

**A.** `Color:`

**B.** `Color:null`

**C.** `Color:White`

**D.** Compiler error on line 4.

**E.** Compiler error on line 10.

**F.** Compiler error on line 11.

**13.** What is the output of the following program?

```
1: public class BearOrShark {
2:   public static void main(String[] args) {
3:     int luck = 10;
4:     if((luck>10 ? luck++: --luck)<10) {
5:       System.out.print("Bear");
6:     } if(luck<10) System.out.print("Shark");
7: } }
```

**A.** `Bear`

**B.** `Shark`

**C.** `BearShark`

**D.** The code will not compile because of line 4.

**E.** The code will not compile because of line 6.

**F.** The code compiles without issue but does not produce any output.

**14.** Assuming we have a valid, non-`null` `HenHouse` object whose value is initialized by the blank line shown here, which of the following are possible outputs of this application? (Choose all that apply)

```
1: class Chicken {}
2: interface HenHouse { public java.util.List<Chicken> getChickens(); }
3: public class ChickenSong {
```

```
4:   public static void main(String[] args) {
5:     HenHouse house = _______________
6:     Chicken chicken = house.getChickens().get(0);
7:     for(int i=0; i<house.getChickens().size();
8:       chicken = house.getChickens().get(i++)) {
9:       System.out.println("Cluck");
10: } } }
```

**A.** The code will not compile because of line 6.

**B.** The code will not compile because of lines 7–8.

**C.** The application will compile but not produce any output.

**D.** The application will output `Cluck` exactly once.

**E.** The application will output `Cluck` more than once.

**F.** The application will compile but produce an exception at runtime.

**15.** Which of the following statements can be inserted in the blank line so that the code will compile successfully? (Choose all that apply)

```
public interface CanSwim {}
public class Amphibian implements CanSwim {}
class Tadpole extends Amphibian {}
public class FindAllTadPole {
  public static void main(String[] args) {
    List<Tadpole> tadpoles = new ArrayList<Tadpole>();
    for(Amphibian amphibian : tadpoles) {
      ___________ tadpole = amphibian;
} } }
```

**A.** `CanSwim`

**B.** `Long`

**C.** `Amphibian`

**D.** `Tadpole`

**E.** `Object`

**16.** What individual changes, if any, would allow the following code to compile? (Choose all that apply)

```
1: public interface Animal { public default String getName() { return null; } }
2: interface Mammal { public default String getName() { return null; } }
3: abstract class Otter implements Mammal, Animal {}
```

**A.** The code compiles without issue.

**B.** Remove the `default` method modifier and method implementation on line 1.

**C.** Remove the `default` method modifier and method implementation on line 2.

**D.** Remove the `default` method modifier and method implementation on lines 1 and 2.

**E.** Change the return value on line 1 from `null` to `"Animal"`.