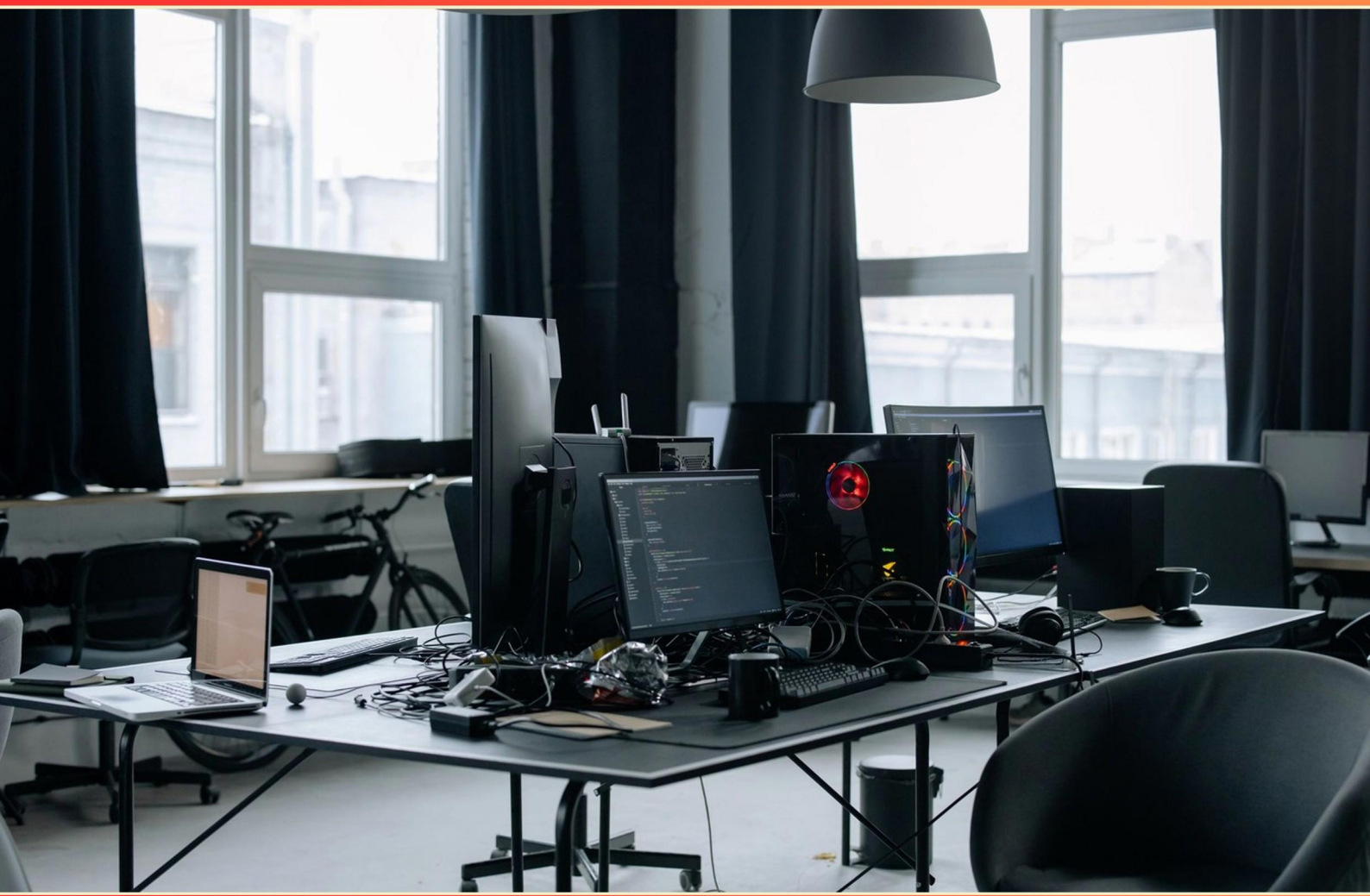
JAVA FPT SOFTWARE (FSOFT) ACADEMY PRACTICE TEST (SAMPLE)

STUDY GUIDE



SAMPLE STUDY GUIDE WITH LIMITED QUESTIONS

VISIT US ONLINE FOR THE FULL VERSION STUDY GUIDE

https://javafsoftacademy.examzify.com

Copyright © 2026 by Examzify - A Kaluba Technologies Inc. product.

ALL RIGHTS RESERVED.

No part of this book may be reproduced or transferred in any form or by any means, graphic, electronic, or mechanical, including photocopying, recording, web distribution, taping, or by any information storage retrieval system, without the written permission of the author.

Notice: Examzify makes every reasonable effort to obtain accurate, complete, and timely information about this product from reliable sources.

SAMPLE

Table of Contents

Copyright 1

Table of Contents 2

Introduction 3

How to Use This Guide 4

Questions 5

Answers 8

Explanations 10

Next Steps 15

SAMPLE

Introduction

Preparing for a certification exam can feel overwhelming, but with the right tools, it becomes an opportunity to build confidence, sharpen your skills, and move one step closer to your goals. At Examzify, we believe that effective exam preparation isn't just about memorization, it's about understanding the material, identifying knowledge gaps, and building the test-taking strategies that lead to success.

This guide was designed to help you do exactly that.

Whether you're preparing for a licensing exam, professional certification, or entry-level qualification, this book offers structured practice to reinforce key concepts. You'll find a wide range of multiple-choice questions, each followed by clear explanations to help you understand not just the right answer, but why it's correct.

The content in this guide is based on real-world exam objectives and aligned with the types of questions and topics commonly found on official tests. It's ideal for learners who want to:

- Practice answering questions under realistic conditions,
- Improve accuracy and speed,
- Review explanations to strengthen weak areas, and
- Approach the exam with greater confidence.

We recommend using this book not as a stand-alone study tool, but alongside other resources like flashcards, textbooks, or hands-on training. For best results, we recommend working through each question, reflecting on the explanation provided, and revisiting the topics that challenge you most.

Remember: successful test preparation isn't about getting every question right the first time, it's about learning from your mistakes and improving over time. Stay focused, trust the process, and know that every page you turn brings you closer to success.

Let's begin.

SAMPLE

How to Use This Guide

This guide is designed to help you study more effectively and approach your exam with confidence. Whether you're reviewing for the first time or doing a final refresh, here's how to get the most out of your Examzify study guide:

1. Start with a Diagnostic Review

Skim through the questions to get a sense of what you know and what you need to focus on. Your goal is to identify knowledge gaps early.

2. Study in Short, Focused Sessions

Break your study time into manageable blocks (e.g. 30 – 45 minutes). Review a handful of questions, reflect on the explanations.

3. Learn from the Explanations

After answering a question, always read the explanation, even if you got it right. It reinforces key points, corrects misunderstandings, and teaches subtle distinctions between similar answers.

4. Track Your Progress

Use bookmarks or notes (if reading digitally) to mark difficult questions. Revisit these regularly and track improvements over time.

5. Simulate the Real Exam

Once you're comfortable, try taking a full set of questions without pausing. Set a timer and simulate test-day conditions to build confidence and time management skills.

6. Repeat and Review

Don't just study once, repetition builds retention. Re-attempt questions after a few days and revisit explanations to reinforce learning. Pair this guide with other Examzify tools like flashcards, and digital practice tests to strengthen your preparation across formats.

There's no single right way to study, but consistent, thoughtful effort always wins. Use this guide flexibly, adapt the tips above to fit your pace and learning style. You've got this!

SAMPLE

Questions

SAMPLE

1. **In the example where a static integer a is initialized to 40, what is printed by the program?**

   A. 40 40

   B. 40 80

   C. 0 80

   D. 0 40

2. **Which statement is true about primitive types and wrapper classes in the context of autoboxing?**

   A. Primitives are objects; wrappers are not.

   B. Wrapper classes are immutable objects; primitives are not objects.

   C. Autoboxing converts wrappers to primitives.

   D. Null can be assigned to a primitive variable.

3. **What happens if you try to instantiate an abstract class?**

   A. It compiles and runs.

   B. It compiles but throws runtime error.

   C. It does not compile.

   D. It prints an error at runtime.

4. **If b starts as 8 and c starts as 3, what is the value of b after performing b = (byte) (b ^ c)?**

   A. 8

   B. 11

   C. 12

   D. 3

5. **Which feature of java.util.concurrent Locks allows configuring thread acquisition order?**

   A. Fairness policy.

   B. Reentrancy level.

   C. Spin-wait mode.

   D. Blocking pattern.

SAMPLE

6. In a two-dimensional array example where the inner loop breaks on the value 2, what is the printed sequence?

A. 1 2 3 4 5

B. 1 2 4 5

C. 1 2

D. 1 4 5

7. Which technique is commonly used to perform a deep copy by converting an object graph to a byte stream and back?

A. Cloneable interface

B. Copy constructor

C. Manual field-by-field copying

D. Serialization

8. Which expression creates a String at compile time due to constant folding?

A. "ab" + "cd"

B. new String("ab" + "cd")

C. "ab" + new String("cd")

D. new String("abcd")

9. What is the value of a[3] for the array int[] a = {1, 2, 3, 4};

A. 1

B. 2

C. 3

D. 4

10. Consider the code snippet: int arr[2]; System.out.println(arr[0]); System.out.println(arr[1]); What happens when compiling and running this code in Java?

A. 0 and 0

B. garbage value and garbage value

C. Compiler Error

D. Runtime error

SAMPLE

Answers

SAMPLE

1. A
2. B
3. C
4. B
5. A
6. B
7. D
8. A
9. D
10. C

SAMPLE

Explanations

SAMPLE

1. In the example where a static integer a is initialized to 40, what is printed by the program?

A. 40 40

B. 40 80

C. 0 80

D. 0 40

Static fields belong to the class itself, not to any particular object. They are initialized once when the class is loaded and retain their value until you explicitly change it. In this example, the static integer starts at 40 and, if nothing changes between the two print statements, both prints will show the same value. Since a is not modified between the prints, the output is 40 40. If there were a statement that changed a between the prints, you could see something like 40 80; or if a were shadowed or reset, other results could appear. But with no modification, the value stays 40 for both prints.

2. Which statement is true about primitive types and wrapper classes in the context of autoboxing?

A. Primitives are objects; wrappers are not.

B. Wrapper classes are immutable objects; primitives are not objects.

C. Autoboxing converts wrappers to primitives.

D. Null can be assigned to a primitive variable.

In Java, primitive values are raw data, not objects, while wrapper classes wrap those values in immutable objects. Autoboxing automatically converts a primitive to its wrapper (for example, int to Integer) and unboxing does the reverse. Because primitives aren't objects, they can't be null, whereas wrappers are objects that can hold a value or be null. This is why the statement that wrapper classes are immutable objects and primitives are not objects is the correct description of how autoboxing relates primitives and wrappers. The other options don't fit: primitives aren't objects, autoboxing goes from primitive to wrapper (not wrapper to primitive), and a primitive variable cannot be null.

3. What happens if you try to instantiate an abstract class?

A. It compiles and runs.

B. It compiles but throws runtime error.

C. It does not compile.

D. It prints an error at runtime.

Abstract classes are blueprints for other classes. You cannot create an object of an abstract class because it may declare methods without implementations. The compiler enforces this and reports a compile-time error, so the code that tries to instantiate such a class won't even run. To use the class's functionality, you must define a concrete subclass that implements all abstract methods and instantiate that subclass. You could also create an anonymous subclass on the spot that provides the necessary implementations. In short, trying to instantiate an abstract class leads to a compile-time failure, not a runtime one.

SAMPLE

4. If b starts as 8 and c starts as 3, what is the value of b after performing b = (byte) (b ^ c)?

A. 8

B. 11

C. 12

D. 3

Bitwise XOR compares each bit and yields 1 where the bits differ. Here, b is 8 (00001000) and c is 3 (00000011). When applying the XOR operator, Java promotes them to int, so 8 ^ 3 equals 11 (00001011). Casting that back to a byte doesn't change the value, so b becomes 11. This is why the result is 11.

5. Which feature of java.util.concurrent Locks allows configuring thread acquisition order?

A. Fairness policy.

B. Reentrancy level.

C. Spin-wait mode.

D. Blocking pattern.

Configuring which thread gets the lock next is controlled by the fairness policy of the lock. In Java's concurrent locks, you can choose a fair lock or a nonfair lock. A fair lock uses a first-come, first-served queue, so threads acquire the lock in the order they requested it, preventing newer requests from jumping ahead of waiting ones. This behavior is set when you construct a lock (for example, new ReentrantLock(true) creates a fair lock). In contrast, a nonfair lock does not guarantee any specific ordering and may let a thread that arrives later acquire the lock sooner, which can boost throughput but risks starvation for waiting threads. So the feature being asked about is the fairness policy. Reentrancy level is about a thread re-entering a lock it already holds, spin-wait mode isn't a standard lock configuration in this context, and the blocking pattern describes how threads wait rather than the order in which they acquire the lock.

6. In a two-dimensional array example where the inner loop breaks on the value 2, what is the printed sequence?

A. 1 2 3 4 5

B. 1 2 4 5

C. 1 2

D. 1 4 5

The behavior being tested is how break works inside nested loops. Break only exits the inner loop, so the outer loop continues to process the next row. If you print elements as you traverse and break when you encounter a 2, you'll print the 2 in that row and then move on, not continuing with the rest of that row. In this example, the first row prints 1 and 2, then stops before reaching 3. The second row starts and prints 4 and 5. Put together, the printed sequence is 1 2 4 5.

SAMPLE

7. Which technique is commonly used to perform a deep copy by converting an object graph to a byte stream and back?

A. Cloneable interface

B. Copy constructor

C. Manual field-by-field copying

D. Serialization

When you need a true deep copy of an entire object graph, serialization—converting the object into a byte stream and then reconstructing it from that stream—is commonly used. This approach walks through the whole graph, writes the state of every reachable object, and then creates new instances during deserialization, preserving the overall structure and the relationships between objects. Because the entire graph is serialized and then rebuilt, nested objects are copied automatically without writing custom copy logic, making it a natural choice for deep copies. Other options require explicit, manual work to achieve deep copying. The clone approach (via a cloning interface) typically provides shallow copies unless you implement deep copying yourself for every nested object, which is error-prone. A copy constructor or manual field-by-field copying similarly demands you write and maintain all the deep-copy logic yourself, duplicating effort whenever the object graph changes. Keep in mind that serialization needs all involved classes to implement the Serializable contract, and there are considerations like transient fields, versioning, and potential performance overhead. But for a straightforward deep copy of an entire graph, serialization is the go-to technique.

8. Which expression creates a String at compile time due to constant folding?

A. "ab" + "cd"

B. new String("ab" + "cd")

C. "ab" + new String("cd")

D. new String("abcd")

Constant folding happens when the compiler evaluates expressions with known constants. For strings, Java folds concatenations of literal values at compile time, turning "ab" + "cd" into the single literal "abcd" directly in the code. That means no runtime object is created from this expression; the string value is determined during compilation. The other forms involve creating a new String object at runtime (either explicitly with new String(...) or because one side isn't a compile-time constant), so they do not produce a compile-time constant string. For example, wrapping a concatenation in new String(...) forces a new String instance to be created at runtime, and mixing a literal with a non-literal prevents compile-time folding as well. Wrapping a literal in new String(...) also results in a runtime object creation rather than a compile-time constant.

9. What is the value of a[3] for the array int[] a = {1, 2, 3, 4};

A. 1

B. 2

C. 3

D. 4

In Java, arrays use zero-based indexing, so the index 0 refers to the first element. For the array {1, 2, 3, 4}, the elements sit at indices 0, 1, 2, and 3 respectively. That means index 3 corresponds to the fourth element, which is 4. Therefore, a[3] is 4. (If you tried a[4], it would be out of bounds and cause an exception.)

SAMPLE

10. Consider the code snippet: int arr[2]; System.out.println(arr[0]); System.out.println(arr[1]); What happens when compiling and running this code in Java?

A. 0 and 0

B. garbage value and garbage value

C. Compiler Error

D. Runtime error

In Java, arrays are declared with the type followed by brackets and must be created using new, like int[] arr = new int[2]. The syntax shown, int arr[2];, is not valid Java—this style comes from C/C++. Because of that, the code won't even compile, so the outcome is a Compiler Error at compile time. If you corrected the declaration to int[] arr = new int[2];, printing arr[0] and arr[1] would output 0 and 0, since elements of a newly created int array default to 0 in Java.

SAMPLE

Next Steps

Congratulations on reaching the final section of this guide. You've taken a meaningful step toward passing your certification exam and advancing your career.

As you continue preparing, remember that consistent practice, review, and self-reflection are key to success. Make time to revisit difficult topics, simulate exam conditions, and track your progress along the way.

If you need help, have suggestions, or want to share feedback, we'd love to hear from you. Reach out to our team at hello@examzify.com.

Or visit your dedicated course page for more study tools and resources:

https://javafsoftacademy.examzify.com

We wish you the very best on your exam journey. You've got this!

SAMPLE