oraclestudy.com
#1 Oracle Certification Guide

1Z0-851

Java SE 6 Programmer Certified Professional

Exam Summary – Syllabus – Questions

Table of Contents

Introduction to 1Z0-851 Exam on Java SE 6 Programmer Certified Professional

You can use this document to collect all the information about Java SE 6 Programmer Certified Professional (1Z0-851) certification. The Oracle 1Z0-851 certification is mainly targeted to those candidates who are from enterprise software development background and want to flourish their career with Oracle Certified Professional Java SE 6 Programmer (OCPJP) credential. The Java SE 6 Programmer Certified Professional certification exam validates your understanding of the Oracle Java technology and sets the stage for your future progression.

Oracle 1Z0-851 Certification Details:

Exam Name	Java SE 6 Programmer Certified Professional
Exam Code	1Z0-851
Exam Product Version	Java SE
Exam Price	USD $245 (Pricing may vary by country or by localized currency)
Duration	150 Mins
Number of Questions	60
Passing Score	61%
Validated Against	This exam has been validated against SE 5 and SE 6.
Format	Multiple Choice
Recommended Training	Java Programming Language, Java SE 6 (self-study) Fundamentals of the Java Programming Language, Java SE 6 (self-study)
Schedule Exam	Pearson VUE - Oracle
Recommended Practice	**1Z0-851 Online Practice Exam**

Oracle 1Z0-851 Exam Syllabus:

Section 1: Declarations, Initialization and Scoping	- Develop code that declares classes (including abstract and all forms of nested classes), interfaces, and enums, and includes the appropriate use of package and import statements (including static imports). - Develop code that declares an interface. Develop code that implements or extends one or more interfaces. - Develop code that declares an abstract class. Develop code that extends an abstract class. - Develop code that declares, initializes, and uses primitives, arrays, enums, and objects as static, instance, and local variables. Also, use legal identifiers for variable names. - Given a code example, determine if a method is correctly overriding or overloading another method, and identify legal return values (including covariant returns), for the method. - Given a set of classes and superclasses, develop constructors for one or more of the classes. Given a class declaration, determine if a default constructor will be created, and if so, determine the behavior of that constructor. Given a nested or non-nested class listing, write code to instantiate the class.
Section 2: Flow Control	- Develop code that implements an if or switch statement; and identify legal argument types for these statements. - loop (for-each), do, while, labels, break, and continue; and explain the values taken by loop counter variables during and after loop execution. - Develop code that makes use of assertions, and distinguish appropriate from inappropriate uses of assertions. - Develop code that makes use of exceptions and exception handling clauses (try, catch, finally), and declares methods and overriding methods that throw exceptions. - Recognize the effect of an exception arising at a specified point in a code fragment. Note that the exception may be a runtime exception, a checked exception, or an error. - Recognize situations that will result in any of the following being thrown: ArrayIndexOutOfBoundsException,ClassCastException, IllegalArgumentException, IllegalStateException, NullPointerException, NumberFormatException, AssertionError, ExceptionInInitializerError, StackOverflowError or NoClassDefFoundError. Understand which of these are thrown by the virtual machine and recognize situations in which others should be thrown programatically.

Section 3: API Contents	- Develop code that uses the primitive wrapper classes (such as Boolean, Character, Double, Integer, etc.), and/or autoboxing & unboxing. Discuss the differences between the String, StringBuilder, and StringBuffer classes. - Given a scenario involving navigating file systems, reading from files, writing to files, or interacting with the user, develop the correct solution using the following classes (sometimes in combination), from java.io: BufferedReader, BufferedWriter, File, FileReader, FileWriter, PrintWriter, and Console. - Use standard J2SE APIs in the java.text package to correctly format or parse dates, numbers, and currency values for a specific locale; and, given a scenario, determine the appropriate methods to use if you want to use the default locale or a specific locale. Describe the purpose and use of the java.util.Locale class. - Write code that uses standard J2SE APIs in the java.util and java.util.regex packages to format or parse strings or streams. For strings, write code that uses the Pattern and Matcher classes and the String.split method. Recognize and use regular expression patterns for matching (limited to: . (dot), * (star), + (plus), ?, \d, \s, \w, [], ()). The use of *, +, and ? will be limited to greedy quantifiers, and the parenthesis operator will only be used as a grouping mechanism, not for capturing content during matching. For streams, write code using the Formatter and Scanner classes and the PrintWriter.format/printf methods. Recognize and use formatting parameters (limited to: %b, %c, %d, %f, %s) in format strings.
Section 4: Concurrency	- Write code to define, instantiate, and start new threads using both java.lang.Thread and java.lang.Runnable. - Recognize the states in which a thread can exist, and identify ways in which a thread can transition from one state to another. - Given a scenario, write code that makes appropriate use of object locking to protect static or instance variables from concurrent access problems.
Section 5: OO Concepts	- Develop code that implements tight encapsulation, loose coupling, and high cohesion in classes, and describe the benefits. - Given a scenario, develop code that demonstrates the use of polymorphism. Further, determine when casting will be necessary and recognize compiler vs. runtime errors related to object reference casting. - Explain the effect of modifiers on inheritance with respect to constructors, instance or static variables, and instance or static methods. - Given a scenario, develop code that declares and/or invokes overridden or overloaded methods and code that declares and/or invokes superclass, or overloaded

	constructors. - Develop code that implements "is-a" and/or "has-a" relationships.
Section 6: Collections / Generics	- Given a design scenario, determine which collection classes and/or interfaces should be used to properly implement that design, including the use of the Comparable interface. - Distinguish between correct and incorrect overrides of corresponding hashCode and equals methods, and explain the difference between == and the equals method. - Write code that uses the generic versions of the Collections API, in particular, the Set, List, and Map interfaces and implementation classes. Recognize the limitations of the non-generic Collections API and how to refactor code to use the generic versions. Write code that uses the NavigableSet and NavigableMap interfaces. - Develop code that makes proper use of type parameters in class/interface declarations, instance variables, method arguments, and return types; and write generic methods or methods that make use of wildcard types and understand the similarities and differences between these two approaches. - Use capabilities in the java.util package to write code to manipulate a list by sorting, performing a binary search, or converting the list to an array. Use capabilities in the java.util package to write code to manipulate an array by sorting, performing a binary search, or converting the array to a list. Use the java.util.Comparator and java.lang.Comparable interfaces to affect the sorting of lists and arrays. Furthermore, recognize the effect of the "natural ordering" of primitive wrapper classes and java.lang.String on sorting.

Section 7: Fundamentals	- Given a code example and a scenario, write code that uses the appropriate access modifiers, package declarations, and import statements to interact with (through access or inheritance) the code in the example. - Given an example of a class and a command-line, determine the expected runtime behavior. - Determine the effect upon object references and primitive values when they are passed into methods that perform assignments or other modifying operations on the parameters. - Given a code example, recognize the point at which an object becomes eligible for garbage collection, determine what is and is not guaranteed by the garbage collection system, and recognize the behaviors of the Object.finalize() method. - Given the fully-qualified name of a class that is deployed inside and/or outside a JAR file, construct the appropriate directory structure for that class. Given a code example and a classpath, determine whether the classpath will allow the code to compile successfully. - Write code that correctly applies the appropriate operators including assignment operators (limited to: =, +=, -=), arithmetic operators (limited to: +, -, *, /, %, ++, --), relational operators (limited to: <, <=, >, >=, ==, !=), the instanceof operator, logical operators (limited to: &, |, ^, !, &&, ||), and the conditional operator (? :), to produce a desired result. Write code that determines the equality of two objects or two primitives.

1Z0-851 Sample Questions:

01. A programmer has an algorithm that requires a java.util.List that provides an efficient implementation of add(0, object), but does NOT need to support quick random access. What supports these requirements?
a) java.util.Queue
b) java.util.ArrayList
c) java.util.LinearList
d) java.util.LinkedList

02. Given that: Gadget has-a Sprocket and Gadget has-a Spring and Gadget is-a Widget and Widget has-a Sprocket Which two code fragments represent these relationships? (Choose two.)
a) class Widget { Sprocket s; } class Gadget extends Widget { Spring s; }
b) class Widget { } class Gadget extends Widget { Spring s1; Sprocket s2; }
c) class Widget { Sprocket s1; Spring s2; } class Gadget extends Widget {
d) class Gadget { Spring s; } class Widget extends Gadget{ Sprocket s; }
e) class Gadget { } class Widget extends Gadget{ Sprocket s1; Spring s2; }
f) class Gadget { Spring s1; Sprocket s2; } class Widget extends Gadget{ }

03. Given:
2. public class Hi {

3. void m1() { }
4. protected void() m2 { }
5. }
6. class Lois extends Hi {
7. // insert code here
8. }

Which four code fragments, inserted independently at line 7, will compile? (Choose four.)

a) public void m1() { }
b) protected void m1() { }
c) private void m1() { }
d) void m2() { }
e) public void m2() { }
f) protected void m2() { }
g) private void m2() { }

04. Given that t1 is a reference to a live thread, which is true?

a) The Thread.sleep() method can take t1 as an argument.
b) The Object.notify() method can take t1 as an argument.
c) The Thread.yield() method can take t1 as an argument.
d) The Thread.setPriority() method can take t1 as an argument.
e) The Object.notify() method arbitrarily chooses which thread to notify.

05. Given:

1. public class Blip {
2. protected int blipvert(int x) { return 0; }
3. }
4. class Vert extends Blip {
5. // insert code here
6. }

Which five methods, inserted independently at line 5, will compile? (Choose five.)

a) public int blipvert(int x) { return 0; }
b) private int blipvert(int x) { return 0; }
c) private int blipvert(long x) { return 0; }
d) protected long blipvert(int x) { return 0; }
e) protected int blipvert(long x) { return 0; }
f) protected long blipvert(long x) { return 0; }
g) protected long blipvert(int x, int y) { return 0; }

06. Given:

1. interface TestA { String toString(); }
2. public class Test {
3. public static void main(String[] args) {
4. System.out.println(new TestA() {
5. public String toString() { return "test"; }
6. });
7. }
8. }

What is the result?

a) test

b) null
c) An exception is thrown at runtime.
d) Compilation fails because of an error in line 1.
e) Compilation fails because of an error in line 4.
f) Compilation fails because of an error in line 5.

07. A team of programmers is reviewing a proposed API for a new utility class. After some discussion, they realize that they can reduce the number of methods in the API without losing any functionality. If they implement the new design, which two OO principles will they be promoting?
a) Looser coupling
b) Tighter coupling
c) Lower cohesion
d) Higher cohesion
e) Weaker encapsulation
f) Stronger encapsulation

08. Which two statements are true? (Choose two.)
a) It is possible for more than two threads to deadlock at once.
b) The JVM implementation guarantees that multiple threads cannot enter into a deadlocked state.
c) Deadlocked threads release once their sleep() method's sleep duration has expired.
d) Deadlocking can occur only when the wait(), notify(), and notifyAll() methods are used incorrectly.
e) It is possible for a single-threaded application to deadlock if synchronized blocks are used incorrectly.
f) If a piece of code is capable of deadlocking, you cannot eliminate the possibility of deadlocking by inserting invocations of Thread.yield().

09. Given:
```
10. class One {
11. void foo() { }
12. }
13. class Two extends One {
14. //insert method here
15. }
```
Which three methods, inserted individually at line 14, will correctly complete class Two? (Choose three.)
a) int foo() { /* more code here */ }
b) void foo() { /* more code here */ }
c) public void foo() { /* more code here */ }
d) private void foo() { /* more code here */ }
e) protected void foo() { /* more code here */ }

10. Given a class whose instances, when found in a collection of objects, are sorted by using the compareTo() method, which two statements are true? (Choose two.)
a) The class implements java.lang.Comparable.
b) The class implements java.util.Comparator.
c) The interface used to implement sorting allows this class to define only one sort sequence.

d) The interface used to implement sorting allows this class to define many different sort sequences.

Answers to 1Z0-851 Exam Questions:

QUESTION: 01 Answer: d	QUESTION: 02 Answer: a, c	QUESTION: 03 Answer: a, b, e, f	QUESTION: 04 Answer: e	QUESTION: 05 Answer: a, c, e, f, g
QUESTION: 06 Answer: a	QUESTION: 07 Answer: a	QUESTION: 08 Answer: a, f	QUESTION: 09 Answer: b, c, e	QUESTION: 10 Answer: a, c

Note: If you find any typo or data entry error in these sample questions, we request you to update us by commenting on this page or write an email on feedback@oraclestudy.com