


# 1Z0-852

## Java SE 6 Programmer Certified Professional Upgrade

Exam Summary – Syllabus – Questions

# Table of Contents

# Introduction to 1Z0-852 Exam on Java SE 6 Programmer Certified Professional Upgrade

You can use this document to collect all the information about Java SE 6 Programmer Certified Professional Upgrade (1Z0-852) certification. The Oracle 1Z0-852 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 Upgrade certification exam validates your understanding of the Oracle Java technology and sets the stage for your future progression.

## Oracle 1Z0-852 Certification Details:

| | |
|---|---|
| Exam Name | Java SE 6 Programmer Certified Professional Upgrade |
| Exam Code | 1Z0-852 |
| Exam Product Version | Java SE |
| Exam Price | USD $245 (Pricing may vary by country or by localized currency) |
| Duration | 150 Mins |
| Number of Questions | 46 |
| Passing Score | 58% |
| Validated Against | This exam has been validated against SE 6. |
| Format | Multiple Choice |
| Recommended Training | Java Programming Language, Java SE 6 (self-study)<br>Fundamentals of the Java Programming Language, Java SE 6 (self-study) |
| Schedule Exam | Pearson VUE - Oracle |
| Recommended Practice | **1Z0-852 Online Practice Exam** |

# Oracle 1Z0-852 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).<br>- 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.<br>- Develop code that declares both static and non-static methods, and, if appropriate, use method names that adhere to the JavaBeans naming standards. Also develop code that declares and uses a variable-length argument list.<br>- 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.<br>- 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.<br>- Develop code that implements all forms of loops and iterators, including the use of for, the enhanced for loop (for-each), do, while, labels, break, and continue; and explain the values taken by loop counter variables during and after loop execution.<br>- Develop code that makes use of assertions, and distinguish appropriate from inappropriate uses of assertions.<br>- Develop code that makes use of exceptions and exception handling clauses (try, catch, finally), and declares methods and overriding methods that throw exceptions.<br>- 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, and so on), and/or autoboxing & unboxing. Discuss the differences between the String, StringBuilder, and StringBuffer classes.<br>- 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.<br>- Develop code that serializes and/or de-serializes objects using the following APIs from java.io: DataInputStream, DataOutputStream, FileInputStream, FileOutputStream, ObjectInputStream, ObjectOutputStream and Serializable.<br>- 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.<br>- 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 | - Recognize the states in which a thread can exist, and identify ways in which a thread can transition from one state to another.<br>- 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.<br>Given a scenario, develop code that demonstrates the use of polymorphism. Further, determine when casting will be necessary and recognize compiler as compared to runtime errors related to object reference casting.<br>- Explain the effect of modifiers on inheritance with respect to constructors, instance or static variables, and instance or static methods. |

| | |
|---|---|
| | - 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.<br>- 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.<br>- 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.<br>- 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 an example of a class and a command-line, determine the expected runtime behavior.<br>- 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. |

# 1Z0-852 Sample Questions:

**01. Given the following six method names:**
addListener addMouseListener setMouseListener deleteMouseListener removeMouseListener registerMouseListener
**How many of these method names follow JavaBean Listener naming rules?**
**a)** 1
**b)** 2
**c)** 3

**d)** 4
**e)** 5

**02. Given that c is a reference to a valid java.io.Console object, which two code fragments read a line of text from the console? (Choose two.)**
**a)** String s = c.readLine();
**b)** char[] c = c.readLine();
**c)** String s = c.readConsole();
**d)** char[] c = c.readConsole();
**e)** String s = c.readLine("%s", "name ");
**f)** char[] c = c.readLine("%s", "name ");

**03. Which can appropriately be thrown by a programmer using Java SE technology to create a desktop application?**
**a)** ClassCastException
**b)** NullPointerException
**c)** NoClassDefFoundError
**d)** NumberFormatException
**e)** ArrayIndexOutOfBoundsException

**04. Which three will compile and run without exception? (Choose three.)**
**a)** private synchronized Object o;
**b)** void go() {
synchronized() { /* code here */ }
}
**c)** public synchronized void go() { /* code here */ }
**d)** private synchronized(this) void go() { /* code here */ }
**e)** void go() {
synchronized(Object.class) { /* code here */ }
}
**f)** void go() {
Object o = new Object();
synchronized(o) { /* code here */ }
}

**05. 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

**06. 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.

**07. Which two code fragments are most likely to cause a StackOverflowError? (Choose two.)**
**a)** int []x = {1,2,3,4,5}; for(int y = 0; y < 6; y++) System.out.println(x[y]);
**b)** static int[] x = {7,6,5,4}; static { x[1] = 8; x[4] = 3; }
**c)** for(int y = 10; y < 10; y++) doStuff(y);
**d)** void doOne(int x) { doTwo(x); } void doTwo(int y) { doThree(y); } void doThree(int z) { doTwo(z); }
**e)** for(int x = 0; x < 1000000000; x++) doStuff(x);
**f)** void counter(int i) { counter(++i); }

**08. Which capability exists only in java.io.FileWriter?**
**a)** Closing an open stream.
**b)** Flushing an open stream.
**c)** Writing to an open stream.
**d)** Writing a line separator to an open stream.

**09. Which two scenarios are NOT safe to replace a StringBuffer object with a StringBuilder object?(Choose two.)**
**a)** When using versions of Java technology earlier than 5.0.
**b)** When sharing a StringBuffer among multiple threads.
**c)** When using the java.io class StringBufferInputStream.
**d)** When you plan to reuse the StringBuffer to build more than one string.

**10. 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.

# Answers to 1Z0-852 Exam Questions:

| | | | | |
|---|---|---|---|---|
| QUESTION: 01<br>Answer: b | QUESTION: 02<br>Answer: a, e | QUESTION: 03<br>Answer: d | QUESTION: 04<br>Answer: c, e, f | QUESTION: 05<br>Answer: d |
| QUESTION: 06<br>Answer: a, c | QUESTION: 07<br>Answer: d, f | QUESTION: 08<br>Answer: d | QUESTION: 09<br>Answer: a, b | QUESTION: 10<br>Answer: e |

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