

<i>Exam Number:</i>	<i>1Z0-804</i>	<i>Duration:</i>	<i>150 minutes</i>
<i>Associated Certifications:</i>	<i><u>Oracle Certified Professional, Java SE 7 Programmer</u></i>	<i>Number of Questions:</i>	<i>90</i>
<i>Exam Product:</i>	<i>Java SE,</i>	<i>Passing Score:</i>	<i>65%</i>
<i>Exam Price:</i>	<i>Rs 8014</i>	<i>Validated Against:</i>	<i>This exam is validated against Java SE 7.</i>
		<i>format:</i>	<i>Multiple Choice</i>

Java Class Design

- Use access modifiers: private, protected, and public
- Override methods
- Overload constructors and methods
- Use the instanceof operator and casting
- Use virtual method invocation
- Override the hashCode, equals, and toString methods from the Object class to improve the functionality of your class.
- Use package and import statements

Advanced Class Design

- Identify when and how to apply abstract classes
- Construct abstract Java classes and subclasses
- Use the static and final keywords
- Create top-level and nested classes
- Use enumerated types

Object-Oriented Design Principles

- Write code that declares, implements and/or extends interfaces
- Choose between interface inheritance and class inheritance
- Apply cohesion, low-coupling, IS-A, and HAS-A principles
- Apply object composition principles (including has-a relationships)
- Design a class using a Singleton design pattern
- Write code to implement the Data Access Object (DAO) pattern
- Design and create objects using a factory pattern

Generics and Collections

- Create a generic class
- Use the diamond for type inference
- Analyze the interoperability of collections that use raw types and generic types
- Use wrapper classes, autoboxing and unboxing
- Create and use List, Set and Deque implementations
- Create and use Map implementations
- Use `java.util.Comparator` and `java.lang.Comparable`
- Sort and search arrays and lists

String Processing

- Search, parse and build strings (including `Scanner`, `StringTokenizer`, `StringBuilder`, `String` and `Formatter`)
- Search, parse, and replace strings by using regular expressions, using expression patterns for matching limited to: `.` (dot), `*` (star), `+` (plus), `?`, `\d`, `\D`, `\s`, `\S`, `\w`, `\W`, `\b`, `\B`, `[]`, `()`.
- Format strings using the formatting parameters: `%b`, `%c`, `%d`, `%f`, and `%s` in format strings.

Exceptions and Assertions

- Use `throw` and `throws` statements
- Develop code that handles multiple Exception types in a single catch block
- Develop code that uses `try-with-resources` statements (including using classes that implement the `AutoCloseable` interface)
- Create custom exceptions
- Test invariants by using assertions

Java I/O Fundamentals

- Read and write data from the console
- Use streams to read from and write to files by using classes in the `java.io` package including `BufferedReader`, `BufferedWriter`, `File`, `FileReader`, `FileWriter`, `DataInputStream`, `DataOutputStream`, `ObjectOutputStream`, `ObjectInputStream`, and `PrintWriter`

Java File I/O (NIO.2)

- Operate on file and directory paths with the Path class
- Check, delete, copy, or move a file or directory with the Files class
- Read and change file and directory attributes, focusing on the BasicFileAttributes, DosFileAttributes, and PosixFileAttributes interfaces
- Recursively access a directory tree using the DirectoryStream and FileVisitor interfaces
- Find a file with the PathMatcher interface
- Watch a directory for changes with the WatchService interface

Building Database Applications with JDBC

- Describe the interfaces that make up the core of the JDBC API (including the Driver, Connection, Statement, and ResultSet interfaces and their relationship to provider implementations)
- Identify the components required to connect to a database using the DriverManager class (including the jdbc URL)
- Submit queries and read results from the database (including creating statements, returning result sets, iterating through the results, and properly closing result sets, statements, and connections)
- Use JDBC transactions (including disabling auto-commit mode, committing and rolling back transactions, and setting and rolling back to savepoints)
- Construct and use RowSet objects using the RowSetProvider class and the RowSetFactory interface
- Create and use PreparedStatement and CallableStatement objects

Threads

- Create and use the Thread class and the Runnable interface
- Manage and control thread lifecycle
- Synchronize thread access to shared data
- Identify code that may not execute correctly in a multi-threaded environment.

Concurrency

- Use collections from the java.util.concurrent package with a focus on the advantages over and differences from the traditional java.util collections.
- Use Lock, ReadWriteLock, and ReentrantLock classes in the java.util.concurrent.locks package to support lock-free thread-safe programming on single variables.
- Use Executor, ExecutorService, Executors, Callable, and Future to execute tasks using thread pools.
- Use the parallel Fork/Join Framework

Localization

- Read and set the locale by using the Locale object
- Build a resource bundle for each locale
- Call a resource bundle from an application
- Format dates, numbers, and currency values for localization with the NumberFormat and DateFormat classes (including number format patterns)
- Describe the advantages of localizing an application
- Define a locale using language and country codes