

Java SE 8 Programmer II

Exam Number: 1Z0-809

Review exam topics

Java Class Design

- Implement encapsulation
- Implement inheritance including visibility modifiers and composition
- Implement polymorphism
- Override hashCode, equals, and toString methods from Object class
- Create and use singleton classes and immutable classes
- Develop code that uses static keyword on initialize blocks, variables, methods, and classes

Advanced Java Class Design

- Develop code that uses abstract classes and methods
- Develop code that uses the final keyword
- Create inner classes including static inner class, local class, nested class, and anonymous inner class
- Use enumerated types including methods, and constructors in an enum type
- Develop code that declares, implements and/or extends interfaces and use the @Override annotation.
- Create and use Lambda expressions

Generics and Collections

- Create and use a generic class
- Create and use ArrayList, TreeSet, TreeMap, and ArrayDeque objects
- Use java.util.Comparator and java.lang.Comparable interfaces
- Collections Streams and Filters
- Iterate using forEach methods of Streams and List
- Describe Stream interface and Stream pipeline
- Filter a collection by using lambda expressions
- Use method references with Streams

Lambda Built-in Functional Interfaces

- Use the built-in interfaces included in the java.util.function package such as Predicate, Consumer, Function, and Supplier
- Develop code that uses primitive versions of functional interfaces
- Develop code that uses binary versions of functional interfaces
- Develop code that uses the UnaryOperator interface

Java Stream API

- Develop code to extract data from an object using peek() and map() methods including primitive versions of the map() method
- Search for data by using search methods of the Stream classes including findFirst, findAny, anyMatch, allMatch, noneMatch
- Develop code that uses the Optional class
- Develop code that uses Stream data methods and calculation methods
- Sort a collection using Stream API
- Save results to a collection using the collect method and group/partition data using the Collectors class
- Use flatMap() methods in the Stream API

Exceptions and Assertions

- Use try-catch and throw statements
- Use catch, multi-catch, and finally clauses
- Use Autoclose resources with a try-with-resources statement
- Create custom exceptions and Auto-closeable resources
- Test invariants by using assertions

Use Java SE 8 Date/Time API

- Create and manage date-based and time-based events including a combination of date and time into a single object using LocalDate, LocalTime, LocalDateTime, Instant, Period, and Duration
- Work with dates and times across timezones and manage changes resulting from daylight savings including Format date and times values
- Define and create and manage date-based and time-based events using Instant, Period, Duration, and TemporalUnit

Java I/O Fundamentals

- Read and write data from the console
- Use BufferedReader, BufferedWriter, File, FileReader, FileWriter, FileInputStream, FileOutputStream, ObjectOutputStream, ObjectInputStream, and PrintWriter in the java.io package.

Java File I/O (NIO.2)

- Use Path interface to operate on file and directory paths
- Use Files class to check, read, delete, copy, move, manage metadata of a file or directory
- Use Stream API with NIO.2

Java Concurrency

- Create worker threads using Runnable, Callable and use an ExecutorService to concurrently execute tasks
- Identify potential threading problems among deadlock, starvation, livelock, and race conditions
- Use synchronized keyword and java.util.concurrent.atomic package to control the order of thread execution
- Use java.util.concurrent collections and classes including CyclicBarrier and CopyOnWriteArrayList
- Use parallel Fork/Join Framework
- Use parallel Streams including reduction, decomposition, merging processes, pipelines and performance.

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

Localization

- Read and set the locale by using the Locale object
- Create and read a Properties file
- Build a resource bundle for each locale and load a resource bundle in an application

Assume the following:

- **Missing package and import statements:** If sample code do not include package or import statements, and the question does not explicitly refer to these missing statements, then assume that all sample code is in the same package, or import statements exist to support them.

- **No file or directory path names for classes:** If a question does not state the file names or directory locations of classes, then assume one of the following, whichever will enable the code to compile and run:
 - All classes are in one file
 - Each class is contained in a separate file, and all files are in one directory
- **Unintended line breaks:** Sample code might have unintended line breaks. If you see a line of code that looks like it has wrapped, and this creates a situation where the wrapping is significant (for example, a quoted String literal has wrapped), assume that the wrapping is an extension of the same line, and the line does not contain a hard carriage return that would cause a compilation failure.
- **Code fragments:** A code fragment is a small section of source code that is presented without its context. Assume that all necessary supporting code exists and that the supporting environment fully supports the correct compilation and execution of the code shown and its omitted environment.
- **Descriptive comments:** Take descriptive comments, such as "setter and getters go here," at face value. Assume that correct code exists, compiles, and runs successfully to create the described effect.