Java SE 6 Programmer Certified Professional Exam

Exam Number:	1Z0-851	**Duration:** 150 minutes
Associated Certifications:	Oracle Certified Professional, Java SE 6 Programmer	**Number of Questions:** 60
		Passing Score: 61%
Exam Product Version:	Java SE	DD towards the exam fees to be drawn in favor of "**ORACLE INDIA PVT LTD"** payable at **BANGALORE**

Exam Topics

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.
- 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.
- 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: Array Index Out Of Bounds Exception, Class Cast Exception, Illegal Argument Exception, Illegal State Exception, Null Pointer Exception, Number Format Exception, Assertion Error, Exception In Initializer Error, Stack Overflow Error or NoClass Def FoundError. 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, String Builder, and String Buffer 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: Buffered Reader, Buffered Writer, File, File Reader, File Writer, Print Writer, 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 obje ct 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

Oracle Database 11*g*: SQL Fundamentals I

Exam Number: 1Z0-051

Associated Certifications: Oracle 11*g* DBA OCA, Oracle 10*g* DBA OCA, Oracle9*i* DBA OCA, Oracle PL/SQL Developer Certified Associate

Duration: 120 minutes

Number of Questions: 70

Passing Score: 60%

Exam Topics

Retrieving Data Using the SQL SELECT Statement

[] List the capabilities of SQL SELECT statements

[] Execute a basic SELECT statement

Restricting and Sorting Data

[] Limit the rows that are retrieved by a query

[] Sort the rows that are retrieved by a query

[] Use ampersand substitution to restrict and sort output at runtime

Using Single-Row Functions to Customize Output

[] Describe various types of functions available in SQL

[] Use character, number, and date functions in SELECT statements

Using Conversion Functions and Conditional Expressions

[] Describe various types of conversion functions that are available in SQL

[] Use the TO_CHAR, TO_NUMBER, and TO_DATE conversion functions

[] Apply conditional expressions in a SELECT statement

Reporting Aggregated Data Using the Group Functions

[] Identify the available group functions

[] Describe the use of group functions

[] Group data by using the GROUP BY clause

[] Include or exclude grouped rows by using the HAVING clause

Displaying Data from Multiple Tables

[] Write SELECT statements to access data from more than one table using equijoins and nonequijoins

[] Join a table to itself by using a self-join

[] View data that generally does not meet a join condition by using outer joins

[] Generate a Cartesian product of all rows from two or more tables

Using Subqueries to Solve Queries

[] Define subqueries

[] Describe the types of problems that the subqueries can solve

[] List the types of subqueries

[] Write single-row and multiple-row subqueries

Using the Set Operators

[] Describe set operators

[] Use a set operator to combine multiple queries into a single query

[] Control the order of rows returned

Manipulating Data

[] Describe each data manipulation language (DML) statement

[] Insert rows into a table

[] Update rows in a table

[] Delete rows from a table

[] Control transactions

Using DDL Statements to Create and Manage Tables

[] Categorize the main database objects

[] Review the table structure

[] List the data types that are available for columns

[] Create a simple table

[] Explain how constraints are created at the time of table creation

[] Describe how schema objects work

Creating Other Schema Objects

[] Create simple and complex views

[] Retrieve data from views

[] Create, maintain, and use sequences

[] Create and maintain indexes

[] Create private and public synonyms

Oracle Database SQL Expert

Exam Number:	1Z0-047
Associated Certifications:	Oracle Database SQL Certified Expert
Exam Product Version:	SQL and PL/SQL,

Duration: 120 minutes

Number of Questions: 70

Passing Score: 66%

DD towards the exam fees to be drawn in favor of "**ORACLE INDIA PVT LTD"** payable at **BANGALORE**

Exam Topics

Retrieving Data Using the SQL SELECT Statement
- List the capabilities of SQL SELECT statements
- Execute a basic SELECT statement
- Describe how schema objects work

Restricting and Sorting Data
- Limit the rows that are retrieved by a query
- Sort the rows that are retrieved by a query

Using Single-Row Functions to Customize Output
- Describe various types of functions that are available in SQL
- Use character, number, and date functions in SELECT statements
- Describe the use of conversion functions

Reporting Aggregated Data Using the Group Functions
- Identify the available group functions
- Describe the use of group functions
- Group data by using the GROUP BY clause
- Include or exclude grouped rows by using the HAVING clause

Displaying Data from Multiple Tables
- Write SELECT statements to access data from more than one table using equijoins and nonequijoins
- Join a table to itself by using a self-join
- View data that generally does not meet a join condition by using outer joins
- Generate a Cartesian product of all rows from two or more tables

Using Subqueries to Solve Queries
- Define subqueries
- Describe the types of problems that subqueries can solve
- List the types of subqueries
- Write single-row and multiple-row subqueries

Using the Set Operators
- Describe set operators
- Use a set operator to combine multiple a single query
- Control the order of rows returned

Manipulating Data
- Describe each data manipulation language (DML) statement
- Insert rows into a table
- Update rows in a table
- Delete rows from a table
- Control transactions

Using DDL Statements to Create and Manage Tables
- Categorize the main database objects
- Review the table structure
- List the data types that are available for columns
- Create a simple table
- Explain how constraints are created at the time of table creation

Creating Other Schema Objects
- Create simple and complex views
- Retrieve data from views
- Create, maintain, and use sequences
- Create and maintain indexes
- Create private and public synonyms

Managing Objects with Data Dictionary Views
- Use the data dictionary views to research data on your objects
- Query various data dictionary views

Controlling User Access
- Differentiate system privileges from object privileges
- Grant privileges on tables
- View privileges in the data dictionary
- Grant roles
- Distinguish between privileges and roles

Managing Schema Objects
- Add constraints
- Create indexes
- Create indexes using the CREATE TABLE statement
- Creating function-based indexes
- Drop columns and set column UNUSED
- Perform FLASHBACK operations
- Create and use external tables

Manipulating Large Data Sets

- Manipulate data using subqueries
- Describe the features of multitable INSERTs
- Use the following types of multitable INSERTs (Unconditional, Conditional and Pivot)
- Merge rows in a table
- Track the changes to data over a period of time

Generating Reports by Grouping Related Data

- Use the ROLLUP operation to produce subtotal values
- Use the CUBE operation to produce crosstabulation values
- Use the GROUPING function to identify the row values created by ROLLUP or CUBE
- Use GROUPING SETS to produce a single result set

Managing Data in Different Time Zones

- Use Various datetime functions

Retrieving Data Using Subqueries

- Write a multiple-column subquery
- Use scalar subqueries in SQL
- Solve problems with correlated subqueries
- Update and delete rows using correlated subqueries
- Use the EXISTS and NOT EXISTS operators
- Use the WITH clause

Hierarchical Retrieval

- Interpret the concept of a hierarchical query
- Create a tree-structured report
- Format hierarchical data
- Exclude branches from the tree structure

Regular Expression Support

- Using Meta Characters
- Regular Expression Functions
- Replacing Patterns
- Regular Expressions and Check Constraints