# Java SE 8 Programmer I Exam

## Java Basics

### Define the scope of variables.

#### Variables

### Define the structure of a Java class.

#### A Closer Look at the "Hello World!" Application

#### Classes

### Create executable Java applications with a main method; run a Java program from the command line; produce console output.

#### "Hello World!" for the NetBeans IDE

#### "Hello World!" for Microsoft Windows

#### "Hello World!" for Solaris OS and Linux

#### A Closer Look at the "Hello World!" Application

### Import other Java packages to make them accessible in your code.

#### Creating and Using Packages

#### Using Package Members

### Compare and contrast the features and components of Java such as: platform independence, object orientation, encapsulation, etc.

#### About the Java Technology

#### Object-Oriented Programming Concepts

## Working with Java Data Types

### Declare and initialize variables (including casting of primitive data types).

#### Variables

#### Initializing Fields

### Differentiate between object reference variables and primitive variables.

#### Primitive Data Types

#### The Numbers Classes

### Know how to read or write to object fields.

#### Inheritance

#### Declaring Member Variables

#### Creating Objects

#### Using Objects

### Explain an object's lifecycle (creation, "dereference by reassignment" and garbage collection).

#### Objects

#### Creating Objects

#### Using Objects

## Using Operators and Decision Constructs

### Use Java operators; use parentheses to override operator precedence.

#### Operators

#### Assignment, Arithmetic, and Unary Operators

#### Equality, Relational, and Conditional Operators

#### Bitwise and Bit Shift Operators

#### Expressions, Statements, and Blocks

### Test equality between strings and other objects using == and equals().

#### Object as a Superclass

### Item 3: Create and use if, if-else, and ternary constructs.

#### The if-then and if-then-else Statements

#### Equality, Relational, and Conditional Operators

### Use a switch statement.

#### The switch Statement

## Creating and Using Arrays

### Declare, instantiate, initialize and use a one-dimensional array.

#### Arrays

### Declare, instantiate, initialize and use a multi-dimensional array.

#### Arrays

## Using Loop Constructs

### Create and use while loops.

#### The while and do-while Statements

### Create and use for loops including the enhanced for loop.

#### The for Statement

### Create and use do-while loops.

#### The while and do-while Statements

### Compare loop constructs.

#### Summary of Control Flow Statements

### Use break and continue.

#### Branching Statements

## Working with Methods and Encapsulation

### Create methods with arguments and return values, including overloaded methods.

#### Returning a Value from a Method

#### Defining Methods

### Apply the static keyword to methods and fields.

#### Variables

#### Understanding Class Members

#### Default Methods

### Create an overloaded method; differentiate between default and user defined constructors.

#### Defining Methods

#### Providing Constructors for Your Classes

### Apply access modifiers.

#### Controlling Access to Members of a Class

### Apply encapsulation principles to a class.

#### Inheritance

#### Inner Class Example

#### Nested Classes

### Determine the effect upon object references and primitive values when they are passed into methods that change the values.

#### Passing Information to a Method or a Constructor

## Working with Inheritance

### Describe inheritance and its benefits.

#### Inheritance

#### Overriding and Hiding Methods

### Develop code that makes use of polymorphism; develop code that overrides methods; differentiate between the type of a reference and the type of an object.

#### Polymorphism

#### Creating Objects

#### Using Objects

### Determine when casting is necessary.

#### Inheritance

### Use super and this to access objects and constructors.

#### Using the Keyword super

#### Using the this Keyword

### Use abstract classes and interfaces.

#### Abstract Methods and Classes

#### Defining an Interface

#### Implementing an Interface

## Handling Exceptions

### Differentiate among checked exceptions, RuntimeException, and Error.

#### The Catch or Specify Requirement

### Create a try-catch block and determine how exceptions alter normal program flow.

#### Catching and Handling Exceptions

#### The try Block

#### The catch Blocks

### Describe the advantages of exception handling .

#### What Is an Exception?

#### Advantages of Exceptions

### Create and invoke a method that throws an exception.

#### Catching Exceptions

### Recognize common exception classes and categories (such as NullPointerException, ArithmeticException, ArrayIndexOutOfBoundsException, ClassCastException).

## Working with Selected classes from the Java API

### Manipulate data using the StringBuilder class and its methods.

#### The StringBuilder Class

#### Summary of Characters and Strings

### Create and manipulate strings.

#### Strings

#### Converting Between Numbers and Strings

#### Comparing Strings and Portions of Strings

#### Manipulating Characters in a String

### Create and manipulate calendar data using classes from java.time.LocalDateTime, java.time.LocalDate, java.time.LocalTime, java.time.format.DateTimeFormatter, java.time.Period.

#### Date and Time Classes

#### Date Classes

#### Parsing and Formatting

#### Period and Duration

### Declare and use an ArrayList of a given type.

#### The List Interface

#### List Implementations

### Write a simple Lambda expression that consumes a Lambda Predicate expression.

#### Lambda Expressions

#### Aggregate Operations