Glossary

Abstract class An abstract class is a type of class that is not allowed to be instantiated. The only reason it exists is to be extended. Abstract classes contain methods and variables common to all the subclasses, but the abstract class itself is of a type that will not be used directly. Even a single abstract method requires that a class be marked `abstract`.

Abstract method An abstract method is a method declaration that contains no functional code. The reason for using an abstract method is to ensure that subclasses of this class will include an implementation of this method. Any concrete class (that is, a class that is not abstract, and therefore capable of being instantiated) must implement all abstract methods it has inherited.

Access modifier An access modifier is a modifier that changes the visibility of a class or member of a class (method, variable, or nested class).

Anonymous inner classes Anonymous inner classes are local inner classes that do not have a class name. You create an anonymous inner class by creating an instance of a new unnamed class that is either a subclass of a named class type or an implementer of a named interface type.

API Application programmers interface. This term refers to a set of related classes and methods that work together to provide a particular capability. The API represents the parts of a class that are exposed (through access controls) to code written by others.

Array Arrays are homogenous data structures implemented in Java as objects. Arrays store one or more of a specific type and provide indexed access to the store.

Automatic variables Also called method local or stack variables. Automatic variables are variables that are declared within a method and discarded when the method has completed.

Base class A base class is a class that has been extended. If class D extends class B, class B is the base class of class D.

Blocked state A thread that is waiting for a resource, such as a lock, to become available is said to be in a blocked state. Blocked threads consume no processor resources.

Boolean expression An expression that results in a value of *true* or *false*. Typically, this involves a comparison (e.g., x > 2) or a boolean condition such as (x < 5 && y > 3), but can also involve a method with a boolean return type.

boolean primitives A primitive *boolean* value can be defined only as either *true* or *false*.

Call stack A call stack is a list of methods that have been called in the order in which they were called. Typically, the most recently called method (the current method) is thought of as being at the top of the call stack.

Casting Casting is the conversion of one type to another type. Typically, casting is used to convert an object reference to either a subtype (for example, casting an Animal reference to a Horse), but casting can also be used on primitive types to convert a larger type to a smaller type, such as from a *long* to an *int*.

char A char is a 16-bit unsigned primitive data type that holds a single Unicode character.

Character literals Character literals are represented by a single character in single quotes, such as 'A'.

Child class *See* Derived class.

Class A class is the definition of a type. It is the blueprint used to construct objects of that type.

Class members Class members are things that belong to a class including methods (static and nonstatic), variables (static and nonstatic), and nested classes (static and nonstatic). Class members can have any of the four access control levels (public, protected, default (package), and private).

Class methods A class method, often referred to as a static method, may be accessed directly from a class, without instantiating the class first.

Class variable *See* static variable.

Collection A collection is an object used to store other objects. Collections are also commonly referred to as containers. Two common examples of collections are HashMap and ArrayList.

Collection interface The collection interface defines the public interface that is common to Set and List collection classes. Map classes (such as HashMap and Hashtable) do not implement Collection, but are still considered part of the collection framework.

Collection framework Three elements (interfaces, implementations, and algorithms) create what is known as the collection framework, and include Sets (which contain no duplicates), Lists (which can be accessed by an index position), and Maps (which can be accessed by a unique identifier).

Comparison operators Comparison operators perform a comparison on two parameters and return a boolean value indicating if the comparison is *true*. For example, the comparison 2<4 will result in *true* while the comparison 4==7 will result in *false*.

Constructor A method-like block of code that is called when the object is created (instantiated) Typically, constructors initialize data members and acquire whatever resources the object may require. It is the code that runs before the object can be referenced.

continue statement The *continue* statement causes the current iteration of the innermost loop to cease and the next iteration of the same loop to start if the condition of the loop is met. In the case of using a *continue* statement with a *for* loop, you need to consider the effects that the continue has on the loop iterator (the iteration expression will run immediately after the *continue* statement).

Deadlock Also called deadly embrace. Threads sometimes block while waiting to get a lock. It is easy to get into a situation where one thread has a lock and wants another lock that is currently owned by another thread that wants the first lock. Deadlock is one of those problems that are difficult to cure, especially because things just stop happening and there are no friendly exception stack traces to study. They might be difficult, or even impossible, to replicate because they always depend on what many threads may be doing at a particular moment in time.

Deadly embrace *See* Deadlock.

Decision statement The *if* and *switch* statements are commonly referred to as decision statements. When you use decision statements in your program, you are asking the program to calculate a given expression to determine which course of action is required.

Declaration A declaration is a statement that declares a class, interface, method, package, or variable in a source file. A declaration can also explicitly initialize a variable by giving it a value.

Default access A class with default access needs no modifier preceding it in the declaration. Default access allows other classes within the same package to have visibility to this class.

Derived class A derived class is a class that extends another class. If class D extends class B, then class D "derives" from class B and is a derived class.

do-while loop The *do-while* loop is slightly different from the *while* statement in that the program execution cycle will always enter the body of a *do-while* at least once. It does adhere to the rule that you do not need brackets around the body if it contains only one statement.

Encapsulation Encapsulation is the process of grouping methods and data together and hiding them behind a public interface. A class demonstrates good encapsulation by protecting variables with private or protected access, while providing more publicly accessible setter and/or getter methods.

Exception Exception has two common meanings in Java. First, an Exception is an object type. Second, an exception is shorthand for "exceptional condition," which is an occurrence that alters the normal flow of an application.

Exception handling Exception handling allows developers to easily detect errors without writing special code to test return values. Better, it lets us handle these errors in code that is nicely separated from the code that generated them and handle an entire class of errors with the same code, and it allows us to let a method defer handling its errors to a previously called method. Exception handling works by transferring execution of a program to an exception handler when an error, or exception, occurs.

Extensibility Extensibility is a term that describes a design or code that can easily be enhanced without being rewritten.

final class The `final` keyword restricts a class from being extended by another class. If you try to extend a final class, the Java compiler will give an error.

final method The `final` keyword applied to a method prevents the method from being overridden by a subclass.

final variables The `final` keyword applied to a variable makes it impossible to reinitialize a variable once it has been assigned a value. For primitives, this means the value may not be altered once it is initialized. For objects, the data within the object may be modified, but the reference variable may not be changed to reference a different object or null.

Finalizer Every class has a special method, called a finalizer, which is called before an object is reclaimed by the Java VM garbage collector. The JVM calls the finalizer for you as appropriate; you never call a finalizer directly. Think of the finalizer as a friendly warning from the virtual machine. Your finalizer should perform two tasks: performing whatever cleanup is appropriate to the object, and calling the superclass finalizer. Finalizers are not guaranteed to be called just because an object becomes eligible for garbage collection, or before a program shuts down, so you should not rely on them.

Floating-point literals Floating-point literals are defined as double by default, but if you want to specify in your code a number as float, you may attach the suffix *F* to the number.

Floating-point numbers Floating-point numbers are defined as a number, a decimal symbol, and more numbers representing the fraction. Unless a method or class is marked with the *strictfp* modifier, floating-point numbers adhere to the IEEE 754 specification.

for loop A *for* loop is used when a program needs to iterate a section of code a known number of times. There are three main parts to a *for* statement. They are the *declaration and initialization* of variables, the *boolean test expression*, and the *iteration expression*. Each of the sections are separated by a semicolon.

Garbage collection The process by which memory allocated to an object that is no longer reachable from a live thread is reclaimed by the Java VM.

Guarded region A section of code within a *try*/*catch* that is watched for errors to be handled by a particular group of handlers.

HashMap class The HashMap class is roughly equivalent to Hashtable, except that it is not synchronized and it permits null values (and one null key) to be stored.

Heap Java manages memory in a structure called a heap. Every object that Java creates is allocated in the heap, which is created at the beginning of the application and managed automatically by Java.

Hexadecimal literals Hexadecimal numbers are constructed using 16 distinct symbols. The symbols used are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

Identifiers Identifiers are names that we assign to classes, methods, and variables. Java is a case-sensitive language, which means identifiers must have consistent capitalization throughout. Identifiers can have letters and numbers, but a number may not begin the identifier name. Most symbols are not allowed, but the dollar sign ($) and underscore (_) symbols are valid. *See also* Reference variable.

if statement An *if* statement tests an expression for a boolean result. This is achieved by using one or more of Java's relational operators (>, ==, etc.) inside the parentheses of the statement to compare two or more variables.

import statement Import statements allow us to refer to classes without having to use a fully qualified name for each class. Import statements do not make classes accessible; all classes in the classpath are accessible. They simply allow you to type the class name in your code rather than the fully qualified (in other words, including the *package*) name.

Inheritance Inheritance is an object-oriented concept that provides for the reuse and modification of an existing type in such a way that many types can be manipulated as a single type. In Java, inheritance is achieved with the `extends` keyword.

Inner classes Inner classes are a type of class and follow most of the same rules as a normal class. The main difference is an inner class is declared within the curly braces of a class or even within a method. Inner classes are also classes defined at a scope smaller than a package. *See also* Anonymous inner classes; Local inner classes; Member inner classes. Static inner classes are not actually inner classes, but are considered top-level *nested* classes.

Instance Once the class is instantiated, it becomes an object. A single object is referred to as an "instance" of the class from which it was instantiated.

Instance variable An instance variable is belongs to an individual object. Instance variables may be accessed from other methods in the class, or from methods in other classes (depending on the access control). Instance variables may not be accessed from static methods, however, because a static method could be invoked when no instances of the class exist. Logically, if no instances exist, then the instance variable will also not exist, and it would be impossible to access the instance variable.

instanceof comparison operator The *instanceof* comparison operator is available for object variables. The purpose of this operator is to determine whether an object is of a given class or interface type (or any of the subtypes of that type).

This comparison may not be made on primitive types and will result in a compile-time error if it is attempted.

Interface An interface defines a group of methods, or a public interface, that must be implemented by any class that implements the interface. An interface allows an object to be treated as a type declared by the interface implemented.

Iterator An iterator provides the necessary behavior to get to each element in a collection without exposing the collection itself. In classes containing and manipulating collections, it is good practice to return an iterator instead of the collection containing the elements you want to iterate over. This shields clients from internal changes to the data structures used in your classes.

Java source file A file that contains computer instructions written in the Java programming language. A Java source file must meet strict requirements; otherwise, the Java compiler will generate errors. Source files must end with a .java extension, and there may be only one public class per source code file.

Java Virtual Machine (JVM) A program that interprets and executes Java bytecode (in most cases, the bytecode that was generated by a Java compiler). The Java VM provides a variety of resources to the applications it is executing, including memory management, network access, hardware abstraction, and so on. Because it provides a consistent environment for Java applications to run in, the Java VM is the heart of the "write once run anywhere" strategy that has made Java so popular.

javac Javac is the name of the java compiler program. This Java compiler processes the source file to produce a bytecode file.

java.lang package The java.lang package defines classes used by all Java programs. The package defines class wrappers for all primitive types such as Boolean, Byte, Character, Double, Float, Integer, Long, and Short, as well as String, Thread, and Object. Unlike classes in any *other* package, classes in the java.lang package may be referred to by just their class name, without having to use an import statement.

JVM *See* Java Virtual Machine.

Keywords Keywords are special reserved words in Java that cannot be used as identifiers for classes, methods, and variables.

Local inner classes You can define inner classes within the scope of a method, or even smaller blocks within a method. We call this a local inner class, and they are often also anonymous classes. Local inner classes cannot use local variables of the method unless those variables are marked final.

Local variable A local variable is a variable declared within a method. These are also known as automatic variables. Local variables, including primitives, must be initialized before you attempt to use them (though not necessarily on the same line of code).

Members Elements of a class, including methods, variables, and nested classes.

Method A section of source code that performs a specific function, has a name, may be passed parameters, and may return a result. Methods are found only within classes.

Method local variables *See* Automatic variables.

Modifier A modifier is a keyword in a class, method, or variable declaration that modifies the behavior of the element. *See also* Access modifier.

notify() method The methods `wait()` and `notify()` are instance methods of an object. In the same way that every object has a lock, every object has a list of threads that are waiting for a signal related to the object. A thread gets on this list by executing the `wait()` method of the object. From that moment, it does not execute any further instructions until some other thread calls the `notify()` method of the same object.

Object Once the class is instantiated it becomes an object (sometimes referred to as an instance).

Overloaded methods Methods are overloaded when there are multiple methods in the same class with the same names but with different parameter lists.

Overridden methods Methods in the parent and subclasses with the same name, parameter list, and return type are overridden.

Package A package is an entity that groups classes together. The name of the package must reflect the directory structure used to store the classes in your package. The subdirectory begins in any directory indicated by the class path environment variable.

Parent class A parent class is a class from which another class is derived. *See also* Base class.

Primitive literal A primitive literal is merely a source code representation of the primitive data types.

Primitives Primitives can be a fundamental instruction, operation, or statement. They must be initialized before you attempt to use them (though not necessarily on the same line of code).

Private members Private members are members of a class that cannot be accessed by any class other than the class in which it is declared.

public access The `public` keyword placed in front of a class allows all classes from all packages to have access to a class.

public members When a method or variable member is declared `public`, it means all other classes, regardless of the package that they belong to, can access the member (assuming the class itself is visible).

Reference The term reference is shorthand for reference variable. *See* Reference variable.

Reference variable A reference variable is an identifier that refers to a primitive type or an object (including an array). A reference variable is a name that points to a location in the computer's memory where the object is stored. A variable declaration is used to assign a variable name to an object or primitive type. A reference variable is a name that is used in Java to reference an instance of a class.

Runtime exceptions A runtime exception is an exception that does not need to be handled in your program. Usually, runtime exceptions indicate a program bug. These are referred to as unchecked exceptions, since the Java compiler does not force the program to handle them.

Shift operators Shift operators shift the bits of a number to the right or left, producing a new number. Shift operators are used on integer types only.

SortedMap interface A data structure that is similar to map except the objects are stored in ascending order according to their keys. Like map, there can be no duplicate keys and the objects themselves may be duplicated. One very important difference with SortedMap objects is that the key may not be a null value.

Source file A source file is a plaintext file containing your Java code. A source file may only have one public class or interface and an unlimited number of default classes or interfaces defined within it, and the filename must be the same as the public class name. *See also* Java source file.

Stack trace If you could print out the state of the call stack at any given time, you would produce a stack trace.

static nested classes Static nested classes are the simplest form of inner classes. They behave much like top-level classes except that they are defined within the scope of another class, namely the enclosing class. Static nested classes have no implicit references to instances of the enclosing class and can access only static members and methods of the enclosing class. Static nested classes are often used to implement small helper classes such as iterators.

static methods The `static` keyword declares a method that belongs to an entire class (as opposed to belonging to an instance). A class method may be accessed directly from a class, without instantiating the class first.

static variable Also called a class variable. A static variable, much like a static method, may be accessed from a class directly, even though the class has not been instantiated. The value of a static variable will be the same in every instance of the class.

String literal A string literal is a source code representation of a value of a string.

String objects An object that provides string manipulation capabilities. The String class is final, and thus may not be subclassed.

Superclass In object technology, a high-level class that passes attributes and methods (data and processing) down the hierarchy to subclasses. A superclass is a class from which one or more other classes are derived.

switch statement The expression in the switch statement can only evaluate to an integral primitive type that can be implicitly cast to an int. These types are byte, short, char, and int. Also, the switch can only check for an equality. This means that the other relational operators like the greater than sign are rendered unusable. *See also* Decision statement.

Synchronized methods The `synchronized` keyword indicates that a method may be accessed by only one thread at a time.

Thread An independent line of execution. The same method may be used in multiple threads. As a thread executes instructions, any variables that it declares within the method (the so-called automatic variables) are stored in a private area of memory, which other threads cannot access. This allows any other thread to execute the same method on the same object at the same time without having its automatic variables unexpectedly modified.

Time-slicing A scheme for scheduling thread execution.

transient variables The `transient` keyword indicates which variables are not to have their data written to an ObjectStream. You will not be required to know anything about *transient* for the exam other than that it is a keyword that can be applied only to variables.

Unchecked exceptions *See* Runtime exceptions.

Variable access Variable access refers to the ability of one class to read or alter (if it is not final) a variable in another class.

Visibility Visibility is the accessibility of methods and instance variables to other classes and packages. When implementing a class, you determine your methods' and instance variables' visibility keywords as `public`, `protected`, `package`, or `default`.