Glossary

Numerics

2d-tree—A *k*-dimensional tree with *k* equal to 2.

2-node—A tree node that contains one data item and has two children. Also known as a *binary node*. See also *3-node* and *4-node*.

2-3 tree—A tree such that each internal node (nonleaf) has either two or three children, and all leaves are at the same level. A node can have a left subtree, a middle subtree, and a right subtree. If a node has two children and contains one data item, the value of the search key in the node must be greater than the value of the search key in the left child and smaller than the value of the search key in the right child. If a node has three children and contains two data items, the value of the smaller search key in the node must be greater than the value of the search key in the left child and smaller than the value of the search key in the middle child; the value of the larger search key in the node must be greater than the value of the search key in the middle child and smaller than the value of the search key in the right child.

2-3-4 tree—Another term for a *2-4 tree*.

2-4 tree—A tree such that each internal node (nonleaf) has either two, three, or four children, and all leaves are at the same level. A node can have a left subtree, a middle-left subtree, a middle-right subtree, and a right subtree. If a node has two or three children, it adheres to the specifications of a 2-3 tree. If a node has four children and three data items, the value of the smaller search key in the node must be greater than the value of the search key in the left child and smaller than the value of the search key in the middle-left child; the value of the middle search key in the node must be greater than the value of the search key in the middle-left child and smaller than the value of the search key in the middle-right child; finally, the value of the larger search key in the node must be greater than the value of the search key in the middle-right child and smaller than the value of the search key in the right child.

3-node—A tree node that contains two data items and has three children. See also *2-node* and *4-node*.

4-node—A tree node that contains three data items and has four children. See also *2-node* and *3-node*.

A

abstract base class—Another term for *abstract class*.

abstract class—A class that is used only as the superclass for other classes. You cannot create objects of an abstract class. Such a class must have the keyword `abstract` in its declaration. Also known as an *abstract base class* or an *abstract superclass*. See also *abstract method*.

abstract data type (ADT)—A specification of data values and the operations on that data. This specification does not indicate how to store the data or how to implement the operations, and it is independent of any programming language.

abstraction—A process that focuses on what a class or method should do instead of how it will accomplish it.

abstract method—A method that has no body. The declaration of such a method must include the keyword `abstract` and end in a semicolon. The class of an abstract method must be abstract.

abstract superclass—Another term for *abstract class*.

access modifier—A Java reserved word—`public`, `private`, or `protected`—included in the header of a class or method or in the declaration of a variable that indicates the availability of the class, method, or variable. Omitting an access modifier specifies *package access*. Also known as a *visibility modifier*.

accessor method—A method that returns the value of an object's data field, or instance variable. Also known as a *get method or getter*.

activation record—A record generated whenever a method is called to document its current state. Included in the record are the values of the method's parameters and local variables, as well as the location of the call to the method. Also called a *frame*.

actor—An element used in software design to represent a human user or a software component that interacts with the proposed software.

acyclic graph—A graph without cycles.

adapter class—A class that provides a restricted interface to another class used in its implementation.

address—A fixed numeric name associated with each byte of memory. See also *reference*.

adjacency list—The n chains of linked nodes that represent a graph of n vertices numbered from 0 to $n - 1$ such that there is a node in the i^{th} chain for vertex j if and only if there is an edge from vertex i to vertex j.

adjacency matrix—An array of n rows and n columns that represents a graph of n vertices numbered from 0 to $n - 1$ such that the entry in row i and column j indicates whether the graph has an edge from vertex i to vertex j.

adjacent vertices—Two vertices of a graph that are joined by an edge. In a directed graph, vertex y is adjacent to vertex x if there is a directed edge from vertex x to vertex y.

ADT—See *abstract data type*.

algorithm—A sequence of steps that solve a problem in a finite amount of time.

alias—A variable that contains the same reference to an object as another variable.

allocate—To assign memory to an object when it is created.

analysis of algorithms—The process of measuring the complexity of algorithms.

ancestor class—Any superclass within a sequence of superclasses that begins when a given class is extended by using inheritance. Every Java class has `Object` as its ancestor. See also *descendant class*.

ancestor of node *n*—A tree node along the path between node n and the root. See also *descendant of node n*.

API—See *Java Application Programming Interface*.

append—(1) To add either characters or a string to the end of a string. (2) To augment a file by adding data to its end.

application programming interface (API)—See *Java Application Programming Interface*.

argument—A value or object that corresponds to a method's parameter and appears within a call to the method. Arguments must correspond to the method's parameters with respect to number, order, and data type.

array—A Java construct that groups items of the same or related data types.

array based—A description of a collection whose implementation uses an array to store its data.

array element—A location in an array. See also *array entry*.

array entry—The value in an element of an array.

array list—An instance of the class `ArrayList`. Such an object maintains its entries in an order determined by its client. See also *list*.

array type—The data type of an array. An array type is a reference type.

array variable—A variable that references and names an array.

ASCII (American Standard Code for Information Interchange)—An integer representation of characters that uses one byte per character. Many programming languages use ASCII, but Java uses Unicode.

assertion—A statement of truth about some aspect of a program's logic. An assertion can be expressed as a comment or by using a Java `assert` statement.

assertion error—The execution-time error that occurs when the boolean expression in an `assert` statement is false and assertions have been enabled.

`assert` statement—A Java statement that enforces an assertion you make in a program. You use such statements during the development of a program, but not as a part of its final logic.

association—A relationship between two objects of different classes. See also *collaboration*.

associative array—Another name for the ADT *dictionary*.

attribute—A property of an object that has a specific value and contributes to the object's state.

average-case time—An estimate of the average time an algorithm requires to solve a problem. See also *best-case time* and *worst-case time*.

AVL tree—A binary search tree that rearranges its nodes whenever it becomes unbalanced due to the insertion or deletion of a node.

B

back of a queue—The end of a queue at which a new entry can be added. See also *front of a queue*.

bag—A collection of items in no particular order. Duplicate items are possible in a bag. See also *set*.

balanced node—A tree node whose subtrees differ in height by no more than 1.

balanced binary tree—A binary tree whose nodes are balanced. That is, every node in a balanced binary tree has subtrees whose heights differ by no more than 1. Also called a *height-balanced tree*.

base case—A smaller problem whose solution is known and must be reached in order for a recursive algorithm to end successfully. Also known as a *stopping case*.

base class—See *superclass*.

basic operation—An operation that is a major contributor to an algorithm's time requirement.

behavior—An action that an object can take.

best-case time—An estimate of the minimum time an algorithm requires to solve a problem. See also *average-case time* and *worst-case time*.

biconnected graph—A graph in which every pair of vertices is joined by two distinct paths that do not share edges or vertices.

bidirectional—A description of an association between two classes that is pictured in a UML class diagram as an arrow pointing in two directions.

Big Oh—A notation that uses the letter O to indicate an algorithm's maximum time requirement. For example, the most time that an $O(n)$ algorithm requires is directly proportional to n, the size of the problem.

Big Omega—A notation that uses the Greek letter Omega (Ω) to indicate an algorithm's minimum time requirement. For example, the least time that an $\Omega(n)$ algorithm requires is directly proportional to n, the size of the problem.

Big Theta—A notation that uses the Greek letter Theta (Θ) to indicate an algorithm's time requirement. For example, the time that a Θ(*n*) algorithm requires is directly proportional to *n*, the size of the problem.

binary file—Any file other than a text file.

binary node—Another name for a 2-node, commonly used to describe the nodes in a binary tree.

binary operator—An operator that acts on two quantities, or operands.

binary search—A way of searching an array that divides the array into two pieces, determines which piece can contain the desired entry, known as the target, and then ignores the other part as it continues the search. See also *interpolation search* and *linear search*.

binary search tree—A binary tree whose nodes contain `Comparable` objects such that each node's data is greater than all the data in the node's left subtree but less than all the data in its right subtree.

binary tree—A tree whose nodes each have no more than two children.

bipartite graph—A graph whose vertices can be divided into two groups such that every edge goes from a vertex in one group to a vertex in the other group.

block—A group of data on a disk.

bound—To restrict a generic type to allow it to have only certain characteristics. Also known as *constrain*.

bounded wildcard—A wildcard used in the definition of a generic data type that is restricted to either a subclass or a superclass of another class. See also *lower bound of a wildcard* and *upper bound of a wildcard*.

breadth-first traversal—A graph-traversal strategy that visits every vertex adjacent to a vertex *v* that it can before it visits any other vertex. Thus, the traversal will not embark from any of the vertices adjacent to *v* until it has visited all possible vertices adjacent to *v*. See also *depth-first traversal*.

B-tree of order *m*—A balanced multiway search tree of order *m* that has the following additional properties: The root has either no children or between 2 and *m* children; each nonleaf has between $[m/2]$ and *m* children; all leaves are on the same level. Typically, a B-tree is stored in an external file.

bubble sort—A way of sorting the entries in an array that involves ordering successive pairs of entries repeatedly until the entire array is sorted.

bucket—(1) A group of values during a radix sort of an array. (2) A portion of a hash table that can contain the group of values whose search keys map to the same hash index.

buffer—A portion of memory that accumulates output before it is sent to the output device.

buffering—The process of accumulating lines of output in a buffer before they are sent to an output device.

C

call by value—A mechanism used to pass an argument to a method whereby the value of the argument is assigned to the argument's corresponding parameter in the method definition. This value is either a primitive value or a reference.

capacity—The maximum number of entries that a particular collection can contain. See also *length of an array.*

cardinality—See *multiplicity*.

catch (an exception)—To react to an exception by means of a `catch` block.

`catch` block—A group of statements preceded by the keyword `catch` that react to an exception. A `catch` block must be associated with a previous `try` block.

`catch` block parameter—An identifier in a `catch` block that represents the thrown exception of a specified type.

ceiling of a number—The smallest integer greater than or equal to a given number. See also *floor of a number*.

chain—A data structure in which each element, or node, contains both data and a reference to another node in the chain. Also called a *singly linked chain*. See also *doubly linked chain*.

checked exception—An exception that is the result of a serious occurrence during program execution. Such an exception belongs to either the standard class `Exception` or one of its descendant classes and must be handled. See also *exception* and *runtime exception*.

child of a node—A tree node that descends directly from a node, known as its parent, at the next level in the tree.

class—A programming construct that describes a group of objects, thereby defining the objects' data type.

class definition—The Java statements that define a class, including its data fields and methods.

class diagram—A diagram in the Unified Modeling Language (UML) to represent a class. It consists of a three-part box that lists a class's name, attributes (data fields), and operations (methods).

class header—The initial portion of a class that occurs before the first open brace. The header gives the class's name and can specify an access modifier, a use modifier, a superclass, and one or more interfaces.

class method—See *static method*.

class-responsibility-collaboration card—See *CRC card*.

class type—The data type of an object other than an array. A class type is the name of the class used to create the object. For example, a string has a data type of `String`. See also *data type* and *reference type*.

class variable—Another term for *static field*.

client—The code that uses a class.

client interface—A term used to collectively describe the headers for all public methods of a class, the comments that tell a programmer how to use these public methods, and any publicly defined constants of the class. See also *Java interface*.

clone—(1) (*n.*) A copy of an object. (2) (*v.*) To make a copy of an object. See also *deep clone* and *shallow clone*.

close—To disconnect a file from its associated stream and the program using it.

clustering—The tendency of items to map into groups, called clusters, of locations in a hash table rather than randomly scattered locations. This difficulty—typical of the linear-probing, collision-resolution scheme in hashing—can cause lengthy search times.

collaboration—An interaction between two classes. See also *association*.

collection—An object that groups other objects, often related by their data type, and provides various services to its client. Typically, a client can add, remove, retrieve, and query the objects in a collection.

collision—A condition that occurs when a hash function maps two or more distinct search keys into the same location of a hash table.

collision-resolution scheme—The part of hashing that assigns locations in the hash table to items with different search keys when the items are involved in a collision. See also *bucket, chain, clustering, double hashing, folding, linear probing, open addressing, probe sequence, quadratic probing,* and *separate chaining*.

color flip—The reversal of a node's color within a red-black tree.

companion classes—A pair of classes whose instances are similar, except that the objects of one class are mutable and the objects of the other class are immutable.

complete binary tree—A binary tree of height h that is full to level $h - 1$ and has level h filled from left to right.

complete graph—A graph that has an edge between every pair of distinct vertices.

completely balanced binary tree—A binary tree in which the left and right subtrees of any node have the same height.

complexity—An algorithm's time and space requirements.

composition—An association, or relationship, between two classes whereby one class has an instance of the other class as a data field. See also *has-a relationship*.

compress—To convert a hash code into an index that lies within a range suitable for a given hash table.

concatenate—To join together, as in joining two strings to form another string.

concordance—An index to a text file of words that indicates the lines in the file containing each word.

connected graph—A graph that has a path between every pair of distinct vertices.

constrain—See *bound*.

core group—A group of core methods.

core method—An essential method that one defines and tests before continuing with the rest of a class definition.

cost—See *weight*.

cost of an algorithm—A general term for an algorithm's complexity.

cost of a path—See *weight of a path*.

counting sort—An algorithm to sort an array of positive integers by counting the number of times each integer occurs within the array.

CRC card—Typically an index card to represent a class during its preliminary design. It lists the class's name, responsibilities, and collaborations with other classes. CRC is an abbreviation for class-responsibility-collaboration.

critical path—The path in a weighted, directed, acyclic graph that has the greatest weight.

cursor—A marker used to track the progress of an iterator. The next entry visited by an iterator will be the one right after the cursor's current position.

cycle—A path in a graph that begins and ends at the same vertex.

D

data abstraction—A design principle that separates the operations that can be performed on a collection of data from the implementation of the operations.

data field—The portion of a class definition that defines an attribute of an object and represents a piece of data. The data field corresponds to an instance variable in an object. Also known as a *data member* or a *field*. See also *instance variable*.

data member—Another name for a *data field*.

data portion (of a node)—The part of a node that contains its data.

data structure—A construct, such as an array, that is defined within a programming language to store data.

data type—A kind of data. A specification of a set of possible values, represented in memory in the same way, and a set of operations on these values. See also *array type, class type, primitive type,* and *reference type*.

decision tree—A tree whose nonleaf nodes contain questions and whose leaves contain answers or conclusions.

deep clone—See *deep copy*.

deep copy—A duplicate of an object that also duplicates all of its component objects. This duplicate is completely distinct from the original object.

delimiter—One or more characters, such as a comma or a space, used to separate one item from another, especially within a string or line of input data.

dense graph—A graph having many edges relative to its number of vertices. See also *sparse graph*.

deprecated—A class or method considered unimportant because of recent changes to a programming language. You should not use a deprecated class or method, as it might be eliminated from the language at a later date.

depth-first traversal—A graph-traversal strategy that proceeds along a path from a given vertex as deeply into the graph as possible before backtracking. That is, after visiting a vertex, the traversal visits, if possible, an unvisited adjacent vertex. If the traversal reaches a vertex that has no unvisited adjacent vertices, it backs up and then visits, if possible, an unvisited adjacent vertex. See also *breadth-first traversal*.

deque—Another term for a *double-ended queue*.

derived class—See *subclass*.

descendant class—Any subclass within a chain of subclasses that begins when a given class is extended by using inheritance. See also *ancestor class*.

descendant of node *n*—A tree node along the path between node *n* and a leaf. See also *ancestor of node n*.

diameter of a graph—The maximum of all the shortest distances between pairs of vertices in an unweighted graph.

dictionary—A data collection whose entries are stored and retrieved according to their search-key values. Also called an *associative array*, a *map*, or a *table*.

dictionary order—See *lexicographic order*.

digraph—See *directed graph*.

direct access—See *random access*.

directed edge—An edge in a directed graph; that is, an edge that has a direction.

directed graph—A graph whose edges indicate a direction. Also called a *digraph*. See also *undirected graph*.

directed path—A sequence of directed edges that begins at one vertex and ends at another vertex in a directed graph. A path in a directed graph. See also *path* and *simple path*.

directly proportional—The relationship between two variables whose values either increase together or decrease together. More precisely, x and y are directly proportional if x equals $c \times y$ for some nonzero constant c.

direct recursion—A form of recursion in which a method contains an explicit call to itself. See also *indirect recursion*.

disconnected graph—A graph that is not connected; that is, a graph that has at least one pair of vertices without a path between them.

divide and conquer—A problem-solving strategy that divides a problem into smaller but distinct problems, each of which is solved separately to produce a solution for the original problem.

double-ended queue—A data collection that is organized like a queue but provides operations on both its oldest and newest entries. Also known as a *deque*.

double hashing—A collision-resolution scheme that uses two hash functions. The hash table is searched for an unoccupied location, starting from the location that one hash function determines and considering every n^{th} location, where n is determined by a second hash function.

doubly linked chain—A linked chain whose nodes each contain two references, one to the next node and one to the previous node. See also *singly linked chain*.

driver—A program that contains a `main` method and that typically tests the methods in a class.

dummy node—In a chain of linked nodes, a first node that is not used for data but is always present. The first data value is actually in the second node.

dynamic allocation—The assignment of memory to a variable during program execution, as opposed to during compilation. See also *static allocation*.

dynamic binding—The mechanism used during program execution to determine which version of a method to call. Also known as *late binding*.

dynamic hashing—A technique in hashing that allows the hash table to grow or shrink in size without the expense of rehashing.

dynamic type—The data type of the object that a variable references at a point during program execution. See also *static type*.

E

edge—The connection between two nodes in a graph or a tree, indicating the relationship between the nodes.

element of an array—See *array element*.

encapsulation—A design principle of object-oriented programming that encloses data and methods within a class, thereby hiding the details of the class's implementation.

enclosing class—Another term for an *outer class*.

entry in an array—See *array entry*.

error—An object of either the standard class `Error` or one of its descendant classes that is created when an abnormal situation, such as running out of memory, occurs during program execution.

error class—The standard class `Error` or one of its descendant classes.

exception—An object of either the standard class `Exception`, the standard class `Error`, or one of their descendant classes that is created when an unusual event or circumstance occurs during program execution. Exceptions are classified as either checked or unchecked.

exception class—The standard class `Exception` or one of its descendant classes.

exclusive or—A boolean operation sometimes used during hashing.

expert system—Software that can assist a user in making a decision or solving a problem.

expression—A combination of operators, variables, literals, and parentheses that represents a value.

extend—To create a subclass from a superclass by using inheritance.

F

fail-safe programming—A technique whereby a programmer includes checks within a program for anticipated errors.

field—Another term for *data field*.

FIFO—See *first-in, first-out*.

file—Data on a particular storage medium, such as a disk. See also *binary file*, *random access*, *sequential access*, and *text file*.

final class—A class that cannot be used as the superclass for any subclass.

`finally` block—A group of statements that can follow either a `try` block or a `catch` block and that must execute regardless of whether an exception occurs.

final method—A method that cannot be overridden by a new definition in a subclass of its class.

first-in, first-out (FIFO)—The behavior exhibited by a data collection such as a queue, whereby the first item added to the collection is the first one that is removed. See also *last-in, first-out (LIFO)*.

floor of a number—The largest integer less than or equal to a given number. See also *ceiling of a number*.

folding—The process of dividing a search key into pieces and then combining the pieces to form a hash code.

frame—Another term for an *activation record*.

front of a queue—The end of a queue at which an entry can be accessed or removed. See also *back of a queue*.

full tree—A binary tree of height *h* whose leaves are all at level *h* and every nonleaf has exactly two children.

fully qualified name—The complete name of a class, including the name of its package. For example, since the class `Scanner` is in the package `java.util`, its fully qualified name is `java.util.Scanner`.

G

game tree—A tree that defines a strategy for game-playing programs.

garbage collection—An automatic process that reclaims memory no longer used by a program and makes it available to the operating system for another purpose.

generalization—A description within the UML of a class's superclass.

general tree—A set of one or more nodes, partitioned into a root node and subsets that are general subtrees of the root.

generic class—A class defined in terms of a generic data type.

generic data type—A placeholder that represents an actual class type within the definition of a class, interface, or static method. Classes that represent collections of objects often use a generic type so that the actual data type of the objects in the collection can be specified when an object of the class is created. Also called a *generic type*, a *generic type parameter*, or a *type parameter*.

generic interface—An interface defined in terms of a generic data type.

generic method—A static method that defines and uses a generic type for the data type of its parameters or return value.

generics—A feature of Java that enables you to write a placeholder—the generic type—instead of an actual class type within the definition of a class or interface.

generic type—Another term for *generic data type*.

generic type parameter—Another term for *generic data type*.

get—To access, typically to access the value of a data field.

get method—Another term for an *accessor method*.

getter—Another term for an *accessor method*.

grammar—A set of rules for forming strings within a language.

graph—A data collection that is a generalization of a tree. A graph focuses on the relationship among its entries instead of any hierarchical organization.

graph coloring—A process that assigns a color to each vertex in a graph such that no two adjacent vertices have the same color. See also *k-colorable*.

graph traversal—A process that starts at vertex *v* and visits all vertices *w* for which there is a path between *v* and *w*. A graph traversal visits every vertex in a graph if and only if the graph is connected, regardless of where the traversal starts.

growth-rate function—A measure of how an algorithm's time requirement grows as the problem size grows. A function of the size of a problem that increases at the same rate as an algorithm's time requirement.

H

handle (an exception)—To detect and react to an exception by using a `try` block, one or more `catch` blocks, and optionally a `finally` block.

***has-a* relationship**—The relationship between a class that has a data field of a class type and the class defining the field's data type. See also *composition*.

hash—To assign, or map, a search key to a hash index.

hash code—An integer that is obtained from a search key during hashing and usually is compressed into a hash index.

hash function—A function that produces a hash index from a given search key by first obtaining the key's hash code and then compressing that code into an index that lies within a given range.

hash index—An integer index into a hash table that is computed from a search key's hash code.

hashing—A technique that locates an entry in an array, called the hash table, using only the entry's search key.

hash table—An array of values corresponding to search keys that is used during hashing.

head reference—An external reference to the first node in a chain of linked nodes. See also *tail reference*.

heap—A complete binary tree whose nodes each contain a `Comparable` value that is no larger (smaller) than the values in the node's children. See also *maxheap* and *minheap*.

height—The number of levels in a tree whose root is at level 1. Alternatively, the number of nodes along the longest path between a tree's root and a leaf.

height-balanced tree—Another term for a *balanced tree*.

hierarchical—A term used to describe a hierarchy. Also known as *nonlinear*.

hierarchy—An arrangement of data into levels, one above the other.

Huffman tree—A binary tree used to generate the Huffman codes used in data compression.

I

immutable—Unchangeable. An immutable object has no methods to change its state, that is, the values of its instance variables. See also *mutable*.

implement—To write the Java definition of a class or a method.

implement an interface—To define the methods within a class that the interface declares.

implementation—A class definition or a method definition.

`implements` clause—The portion of a class header that specifies the interfaces that the class will implement.

`import` statement—A statement that precedes a class definition, indicating the package containing the class or classes used within the class definition.

indirect recursion—A form of recursion whereby a method does not have an explicit call to itself. Instead, it calls a method that calls another, and so on, until the first method is called. For example, method A calls method B, method B calls method C, and method C calls method A. See also *direct recursion* and *mutual recursion*.

induction—See *mathematical induction*.

infinite recursion—A recursion that has no logical ending because it either misses or does not have a base case.

infix expression—An ordinary algebraic expression in which each binary operator appears between its two operands.

information hiding—A technique that involves hiding a class's implementation details from its client.

inheritance—A feature of object-oriented programming that enables you to create a new, more-specialized class based on the definition of an existing general class without repeating the earlier code.

inner class—A nested class that is not static. See also *nested class*.

inner class iterator—An iterator for a data collection that is an instance of a private inner class of the collection's class.

inorder predecessor—The entry in a binary search tree that is visited immediately before the entry currently visited by an inorder traversal. See also *inorder successor*.

inorder successor—The entry in a binary search tree that is visited immediately after the entry currently visited by an inorder traversal. See also *inorder predecessor*.

inorder traversal—A traversal of a binary tree that processes (visits) a node after it traverses the node's left subtree, but before it traverses the node's right subtree. See also *level-order traversal*, *preorder traversal*, and *postorder traversal*.

input stream—A stream that sends data from a source, such as a file or the keyboard, to a program.

insertion sort—A way of sorting the entries in an array that involves dividing, or partitioning, the array into two groups, one of which is sorted and the other unsorted. One by one, an entry in the unsorted portion is inserted into its proper position within the sorted portion until the entire array is sorted.

instance—Another term for an *object*.

instance method—A method that defines a behavior of an object. A method that is not static.

instance variable—A data field within a specific object.

instantiate—To create an object.

instantiation—The process of creating an object.

interface—See *client interface* and *Java interface*.

interface bloat—A characteristic of an abstract data type (ADT) that specifies too many behaviors.

interior node—Another term for a *nonleaf*.

interpolation search—A way of searching an array of sorted and uniformly distributed data. See also *binary search* and *linear search*.

***is-a* relationship**—The relationship between a subclass and its superclass formed by inheritance.

iterate—(1) To traverse the entries in a collection. (2) To cycle through the steps of a process.

iteration—A form of repetition whereby a group of statements is repeated under control of a construct that tests the value of a boolean expression to determine when the repetition should end. See also *recursion*.

iterator—An object that processes, or visits, each entry in a collection.

J

Java Application Programming Interface (API)—Another name for the *Java Class Library*.

Java Class Library—A collection of standard classes provided with the Java language that you can use in your programs. Also known as the *Java Application Programming Interface (API)*.

Java Collections Framework—Interfaces and classes in the Java Class Library that provide a uniform way to process the objects in a collection, regardless of the collection's purpose or implementation.

`javadoc`—A utility program that reads comments written in a particular format and produces HTML documentation that describes public classes and methods.

Java interface—A Java construct containing publicly defined constants and the headers of public methods that a class can define. See also *client interface*.

Java runtime system—The software that supports the execution of an application, including the Java Virtual Machine and the Java Class Library.

Java stack—The program stack in the Java Virtual Machine.

Java Virtual Machine (JVM)—A program that converts Java bytecode into the machine language of its host platform.

jump search—A technique for searching an array that identifies a portion of an array that can contain the target in an attempt to reduce the number of comparisons necessary as compared to a sequential search.

JVM—See *Java Virtual Machine*.

K

***k*-colorable**—The characteristic of a graph that can have *k* or fewer distinct colors assigned to its vertices such that no two adjacent vertices have the same color. See also *graph coloring*.

***k*-dimensional tree**—A binary tree used to organize points in *k*-dimensional space. Also called a *kd-tree*.

***k*d-tree**—Another term for a *k-dimensional tree*.

key—See *search key*.

***k*-node**—A tree node that has $k - 1$ data items and k children.

L

last-in, first-out (LIFO)—The behavior exhibited by a data collection such as a stack, whereby the most recent item added to the collection is the first one that is removed. See also *first-in, first-out (FIFO)*.

late binding—Another term for *dynamic binding*.

leaf—A tree node with no children.

left child of node *n*—A node directly below and to the left of node *n* in a binary tree. See also *right child of node n*.

left-right double rotation—An operation on an AVL tree or a red-black tree that uses a left rotation followed by a right rotation to maintain the tree's balance.

left rotation—An operation on an AVL tree or a red-black tree that maintains the tree's balance.

left subtree of a binary tree—A subtree rooted at the root's left child.

length of an array—The number of elements allocated for an array. The maximum number of entries an array can contain. Also known as the array's *capacity*.

length of a path—The number of edges in a graph's path.

level of a tree node—An indication of a node's hierarchy. The root of a tree is at level 1. The level of any other node is 1 greater than the level of its parent.

level-order traversal—A traversal of a tree that processes (visits) nodes one level at a time, from left to right, beginning with the root. See also *inorder traversal*, *postorder traversal*, and *preorder traversal*.

lexicographic order—The order imposed on characters by their Unicode or ASCII values. Also known as *dictionary order*.

LIFO—See *last-in, first-out*.

linear probing—A collision-resolution scheme that searches the hash table sequentially, starting from the original location specified by the hash function, for an unoccupied location.

linear search—A way of searching an array that examines each array entry in order of occurrence in the array. Also known as a *sequential search*. See also *binary search* and *interpolation search*.

link—The reference in a node of a chain or tree.

linked—The characteristic of two objects such that one or both objects reference the other.

list—A collection of ordered entries that can be referenced by their positions within the collection. See also *array list*.

load factor—A measure of the fullness of a hash table equal to the ratio of the number of entries in a dictionary to the number of locations in the hash table.

loop in a graph—An edge that starts and ends at the same vertex.

lower bound—A value that is less than or equal to all possible values of a given item. See also *upper bound*.

lower bound of a wildcard—Any supertype of a class `C`, as indicated by the expression `? super C`.

M

map—(1) (*n*.) Another name for the ADT *dictionary*. (2) (*v*.) Another word for *hash*.

mathematical induction—A method of proof that establishes the truth of a statement and, based on its truth, proves that the next statement is true.

maxheap—A heap whose nodes each contain a value that is greater than or equal to the values in its descendants. See also *minheap*.

median-of-three pivot selection—A strategy for choosing a pivot during a quick sort by averaging the first, middle, and last entries in the array to be sorted.

member—A data field or method of a class.

memory leak—The failure of an operating system to recycle memory that was once used by a program but is no longer needed. The Java runtime system does not have this fault, as it automatically recycles unneeded memory. See also *garbage collection*.

merge sort—A fast sorting algorithm that involves dividing an array into two pieces, sorting each piece, and then merging the two sorted pieces into a final sorted array.

method—A Java construct that is a part of a class and performs an action. A method defines either a behavior of an object of the class or an action of the class itself. See also *instance method* and *static method*.

method body—The portion of a method definition that follows the method's header and is enclosed in braces.

method declaration—A method's return type followed by its signature and a semicolon. Method declarations appear in a Java interface. See also *method header*.

method definition—The implementation of a method, composed of its header and body.

method header—A method's return type, its name, a pair of parentheses, and optionally, an access modifier, a use modifier, and parameters. See also *method signature*.

method signature—A method's name and the number, order, and types of its parameters.

minheap—A heap whose nodes each contain a value that is less than or equal to the values in its descendants. See also *maxheap*.

multiple edge—Two or more edges that join two vertices in a graph.

multiple inheritance—A form of inheritance that enables a class to have more than one superclass. Although a Java class can have several ancestors, it can have only one superclass. That is, Java does not permit multiple inheritance.

multiplicity—The number of objects involved at each end of an association in a UML class diagram. Also known as *cardinality*.

multiway search tree of order *m*—A general search tree whose nodes have from zero to *m* children each. These nodes can be *k*-nodes for values of *k* ranging from 2 to *m*. Also known as an *m-way search tree*.

mutable—Changeable. A mutable object has one or more methods that can change its state, that is, the values of its instance variables. See also *immutable*.

mutator method—A method that alters the value of an object's instance variable. Also known as a *set method* or *setter*.

mutually exclusive—A description of situations that are disjoint. Only one of several mutually exclusive cases can be true at one time.

mutual recursion—Indirect recursion involving only two methods. For example, method A calls method B, which then calls method A.

***m*-way search tree**—See *multiway search tree of order m*.

N

navigability—An aspect of an association within a UML class diagram, denoted by an arrow head on an end of the association, that indicates a responsibility with respect to the class to which the association points.

neighbor of vertex *v*—Any vertex adjacent to *v*.

nested class—A class defined entirely within another class definition, known as the outer class. See also *inner class*.

node—An object used to form a linked data structure—such as a chain, tree, or graph—that contains both data and a reference, or link, to another node.

nonleaf—A node in a tree that is not a leaf. Also known as an *interior node*.

nonlinear—See *hierarchical*.

O

object—A basic element in a Java program that has values, or attributes—giving it a particular state—and behaviors, some of which can change or report that state. Also known as an *instance*.

object-oriented programming (OOP)—A software engineering technique that views a program as a collection of components called objects that interact. OOP embodies three fundamental principles: encapsulation, inheritance, and polymorphism.

object serialization—A process used by Java to represent an entire object as a sequence of bytes that can be written to a binary file.

object type compatibility—A characteristic of objects that enables you to use an instance of a subclass instead of an instance of a superclass, but not the converse. The object type of an argument in a call to a method can be a descendant of the corresponding parameter's object type.

O(*n*)—A notation, called Big Oh, used to indicate that an algorithm's time requirement is proportional to *n*. See also *Big Oh*, *Big Omega*, and *Big Theta*.

OOP—See *object-oriented programming*.

open—(1) (*adj*.) Available. An open location in a hash table is currently unused. (2) (*v*.) To connect a file to an appropriate stream and associate it with a Java program.

open addressing—A category of collision-resolution schemes in hashing that probe for an empty, or open, location in the hash table in which to place an item. See also *double hashing*, *linear probing*, and *quadratic probing*.

operation—The term used in the Unified Modeling Language (UML) for a method.

origin vertex—The vertex in a graph at which a traversal begins.

outer class—A class that contains the definition of another class. Also known as the *enclosing class*. See also *inner class* and *nested class*.

out of bounds—The condition that occurs when the value of an index is either negative or too large. This condition will cause an exception.

output stream—A stream that sends data from a program to a destination, such as a file or the screen.

overload—To define two or more methods in either the same class or classes related by inheritance that have the same name but different signatures.

override—To define a nonstatic method in a subclass that has the same signature and return type as a method in the superclass.

P

package—A named group of Java classes.

package access—The kind of access that occurs when a class, method, or data field has no access modifier in its header or declaration. Such a class, method, or data field is available only to other classes within its package.

parameter—An identifier within a method's header that is paired with a data type to specify and represent a value or object as input to the method.

parent—A tree node that has one or more nodes that descend directly from it.

parity—The state of an integer with respect to being even or odd.

parse—To analyze syntactically, particularly when referring to a string, input data, or a Java statement.

parse tree—A tree that represents the grammar of a programming language. A parse tree can help a compiler check the syntax of a program.

partition—To divide an array into two or more parts.

path—A sequence of edges in a graph that begins at one vertex and ends at another vertex.

perfect hash function—An ideal hash function that maps each search key into a unique location in the hash table. You can define a perfect hash function when you know all possible search keys.

pivot item—A central element in an algorithm. For example, the quick sort algorithm partitions an array about a particular entry called the pivot.

Polish notation—Another term for the notation used in a prefix expression.

polymorphic variable—A variable having a dynamic type.

polymorphism—A feature of object-oriented programming whereby the correct version of a method is determined during program execution instead of during compilation.

postcondition—A statement of a method's effect after completing its execution. See also *precondition*.

postfix expression—An algebraic expression in which every binary operator follows its two operands. See also *infix expression* and *prefix expression*.

postorder traversal—A traversal of a tree that processes (visits) a node after it traverses each of the node's subtrees. See also *inorder traversal*, *level-order traversal*, and *preorder traversal*.

precedence—Priority. If an operator *a* has precedence over operator *b*, *a* executes before *b*.

precondition—A statement of the requirements necessary for executing a method. See also *postcondition*.

predecessor—(1) In a chain of linked nodes, the predecessor of node *n* is the node that references *n*. (2) In a directed graph, vertex *x* is a predecessor of vertex *y* if there is a directed edge from *x* to *y*, that is, if *y* is adjacent to *x*. See also *successor*.

prefix expression—An algebraic expression in which every binary operator precedes its two operands. See also *infix expression* and *postfix expression*.

preorder traversal—A traversal of a tree that processes (visits) a node before it traverses each of the node's subtrees. See also *inorder traversal*, *level-order traversal*, and *postorder traversal*.

primary clustering—The forming of groups, or clusters, of occupied and consecutive locations within a hash table due to linear probing. See also *secondary clustering.*

prime number—An integer that is divisible by only 1 and itself.

priority queue—A queuelike data collection that organizes its objects according to their priorities instead of chronologically.

private access—The kind of access to a method or data field of a class `C` that enables only `C` to access the method or field by name.

probe sequence—The sequence of locations in the hash table that a collision-resolution scheme examines.

probing—Locating an open, or available, location within a hash table after a collision has occurred.

problem size—The number of items processed by an algorithm as a factor in assessing its time complexity.

program counter—An actual or virtual location that identifies the machine instruction that will execute next.

program stack—The stack within a program's runtime environment that maintains the activation records generated by method execution.

proof by induction—A proof that uses the principle of mathematical induction.

protected access—The kind of access to a method or data field of a class `C` that enables only `C`, a subclass of `C`, or any class in the same package as `C` to access the method or field by name.

pseudocode—A combination of English and Java used to express an algorithm.

pseudorandom number—A calculated number that appears to be chosen at random.

public access—The kind of access to a class, method, or data field that enables any class or method to access it by name.

Q

quadratic probing—A collision-resolution scheme that searches the hash table for an unoccupied location beginning with the original location that the hash function specifies and continuing at increments of 1^2, 2^2, 3^2, and so on.

query method—Another name for an *accessor method*.

queue—A data collection that orders its objects chronologically. The first entry added to a queue is the first one removed. This data organization is known as first-in, first-out, or FIFO.

quick sort—A sorting algorithm that partitions an array's entries around a pivot *p* to generate two smaller sorting problems: Sort the array's left section, whose entries are less than *p*, and sort the array's right section, whose entries are greater than or equal to *p*.

R

radix sort—A sorting algorithm that treats each array entry as a character string and repeatedly organizes the entries into groups according to the i^{th} character in each entry.

random access—An organization of data that enables you to access the data in any desired order. Also called *direct access*. See also *sequential access*.

random number generator—A method or algorithm to produce pseudorandom numbers.

receiver—Another term for *receiving object*.

receiving object—The object whose name occurs in an invocation of one of its methods. Also known as a *receiver*.

recurrence relation—A mathematical formula that generates the terms in a sequence from previous terms.

recursion—A repetitive process that solves a problem by solving smaller and smaller, but otherwise identical, problems. See also *direct recursion*, *indirect recursion*, and *iteration*.

recursive call—An invocation within a method to the method itself. Also known as a *recursive invocation*.

recursive invocation—Another term for *recursive call*.

recursive method—A method that invokes itself.

red-black tree—A representation of a 2-3-4 tree as a binary tree whose nodes have a color, either red or black.

reference—The location in memory of an object.

reference type—Either an array type, a class type, or an interface type. A variable of a reference type contains a reference.

reference variable—A variable of a reference type that contains the location in memory of an object.

rehashing—The process of expanding the size of a hash table when it becomes too full and then computing new hash indices for its contents.

resizing—The process of moving the entries from an array to a new, larger or smaller, array and then giving the new array the name of the original array. The effect is to appear to either expand or reduce the size of the original array.

responsibility—A behavior defined by a class. Those things that an object needs to remember (or keep track of) and those things that an object needs to do for other objects.

reverse Polish notation—Another term for the notation used in a postfix expression.

right child of node *n*—A node directly below and to the right of node *n* in a tree. See also *left child of node n*.

right-left double rotation—An operation on an AVL tree or a red-black tree that uses a right rotation followed by a left rotation to maintain the tree's balance.

right rotation—An operation on an AVL tree or a red-black tree that maintains the tree's balance.

right subtree of a tree—A subtree rooted at the root's right child.

root—The node that is at the top of a tree. The root has no ancestor, but it is the ancestor of all other nodes in the tree.

rotation—An operation used to maintain the balance of an AVL tree or a red-black tree.

runtime exception—An exception that is usually caused by a logical error in the program. Runtime exceptions belong to either the standard class `RuntimeException` or one of its descendants and do not need to be handled. See also *checked exception*, *exception*, and *unchecked exception*.

S

safe and secure programming—A methodology that extends fail-safe programming by validating any input data and arguments to a method by eliminating a method's side effects, and by making no assumptions about the actions of clients and users.

scenario—A description used by class designers to indicate the interaction between an actor and the proposed system. A textual story within a use case that describes a solution to a problem or part of a problem.

scope—The portion of the program in which a variable has meaning.

search—A process that locates a certain item in a collection of items.

search key—The part of an object that identifies it within a collection of objects and is used by a search algorithm to locate it. Also called a *key*.

search tree—A tree whose organization facilitates the search of its data for a particular entry. See also *AVL tree*, *binary search tree*, *B-tree of order m*, *multiway search tree of order m*, *red-black tree*, *2-3 tree*, and *2-3-4 tree*.

secondary clustering—The lengthening of the probe sequence within a hash table due to its repeated use during quadratic probing. See also *primary clustering.*

secure programming—See *safe and secure programming*.

seed—A starting number used when generating pseudorandom numbers.

selection sort—A way of sorting the entries in an array that involves selecting and positioning the smallest (or largest) entry, the next smallest (or next largest) entry, and so on until all entries are sorted.

self-documenting—A term used to describe code that has meaningful names for variables, methods, and classes, making its purpose and logic clear to any programmer who reads it.

semiheap—A complete binary tree in which the root's left and right subtrees are both heaps.

sentinel—A value that ends a data set and is a signal to the program to stop reading.

separate chaining—A collision-resolution scheme that uses an array of linked chains as a hash table. The i^{th} chain contains all items that map into location *i.*

separate class iterator—An iterator for a data collection that is an instance of a public class distinct from the collection's class.

sequence—In mathematics, a finite or infinite list of numbers that have a specific order.

sequential access—An organization of data that requires you to access the data in the order in which is it stored. See also *random access*.

sequential search—See *linear search*.

serialization—See *object serialization*.

set—A collection of distinct items in no particular order. A bag without duplicate entries.

set method—See *mutator method*.

setter—See *mutator method*.

shallow clone—See *shallow copy*.

shallow copy—A duplicate of an object that shares some or all of its component objects with the original object.

Shell sort—An improved insertion sort that moves array entries beyond their adjacent positions to produce a nearly sorted array that is sorted completely by an insertion sort. The total process is faster than an insertion sort on the original array.

shortest path—The path between two given vertices in a weighted graph that has the smallest sum of its edge weights.

siblings—Tree nodes that have a common parent.

side effect—An action taken by a method that is not specified and is secondary to its main purpose.

signature—A method's name and the number, types, and order of its parameters.

silent program—A program that produces no visible output for its user.

simple cycle—A path in a graph that begins and ends at the same vertex and passes through other vertices only once each. See also *cycle*.

simple path—A path in a graph that passes through other vertices only once each. See also *path*.

simulation—A process that imitates a real-world situation by using one or more queues to test various scenarios in business, science, and so on. A technique for modeling the behavior of both natural and artificial systems. Generally, its goal is to generate statistics that summarize the performance of an existing system or to predict the performance of a proposed system. A simulation reflects long-term average behavior of a system rather than predicting occurrences of specific events.

singly linked chain—Another term for a *chain*, especially in the context of or compared with a doubly linked chain.

sort—To arrange items into either ascending or descending order according to some criterion.

sorted list—A data collection that maintains its entries in sorted order and retrieves them by their position number within the list. See also *list*.

space complexity—An algorithm's memory requirements. See also *time complexity*.

sparse graph—A graph having few edges relative to its number of vertices. See also *dense graph*.

sparse hash table—A hash table having few of its locations occupied by data. See also *load factor*.

stable sort—A sorting algorithm that does not change the relative order of objects that are equal. For example, if objects x and y are equal and x appears before y in a collection of data, a stable sorting algorithm will leave object x before object y after sorting the data.

stack—A data collection that organizes its entries chronologically. The last entry added to a stack is the first one removed. This data organization is known as last-in, first out, or LIFO. A stack is used to store the activation records of methods.

state—The collective properties of an object that are defined by its attributes and their values.

static allocation—The assignment of memory to a variable during program compilation. See also *dynamic allocation*.

static field—A data field shared by all objects of a class. Also known as a *class variable* or a *static variable*.

static method—A method that defines an action of a class, but not of any object. Also known as a *class method*.

static type—The data type in a variable's declaration. A static type is fixed and determined during compilation. See also *dynamic type*.

static variable—Another term for *static field*.

stopping case—See *base case*.

stream—An object that represents a flow of data. See also *input stream* and *output stream*.

stream variable—A program variable that references a stream.

string—A sequence of characters enclosed in double quotes. An object of the class `String`.

string literal—A fixed, unnamed value that is a string of characters enclosed in double quotes.

string processing—The manipulation of a string or strings to form another string.

stub—An incomplete method that simply acknowledges that it has been called.

subclass—The class that is the result of extending another class, the superclass. Also known as the *derived class*.

subgraph—A portion of a graph that is itself a graph.

substring—Consecutive characters that make up a portion of a string.

subtree—Any node in a tree, together with all of the node's descendants.

subtree of a node—A subtree whose root is a child of the node.

subtree of a tree—A subtree of the tree's root, and so is rooted at a child of the tree's root. See *subtree of a node*.

subtype—The data type that is a subclass of a given class. See also *supertype*.

successor—(1) In a chain of linked nodes, the successor of node *n* is the node that *n* references. (2) In a directed graph, vertex *y* is a successor of vertex *x* if there is a directed edge from *x* to *y*, that is, if *y* is adjacent to *x*. See also *predecessor*.

superclass—The general class that you extend by using inheritance to define a more-specialized class, the subclass. Also known as the *base class*.

supertype—The data type that is a superclass of a given class. See also *subtype*.

T

table—Another name for the ADT *dictionary*.

tag—An element beginning with `@` that is used in comments to be processed by `javadoc`. A tag identifies such aspects as the programmer's name and each method's parameters, return type, and exceptions it might throw.

tail recursion—A form of recursion in which the last action a method performs is a recursive call.

tail reference—An external reference to the last node in a chain of linked nodes. See also *head reference*.

target—The value to be located when searching an array or data collection.

text file—A file that represents a collection of characters organized as lines. See also *binary file*.

throw an exception—To execute a `throw` statement containing an object of either the class `Exception` or one of its descendant classes.

`throws` clause—The portion of a method header that indicates the exceptions that the method might throw but will not handle.

`throw` statement—The statement that throws a given exception.

tight—Close. A tight approximation is close to the actual value.

time complexity—An algorithm's time requirements. See also *space complexity*.

time-driven simulation—A simulation in which the time of an event, such as an arrival or departure, is determined randomly and compared with a simulated clock.

token—Any group of contiguous characters other than white space.

top-level class—A class that is not nested.

top of a stack—The end of a stack at which an entry can be added, accessed, or removed.

topological order—A list of vertices in a directed graph without cycles such that vertex *x* precedes vertex *y* if there is a directed edge from *x* to *y* in the graph. A topological order is not unique, in general.

topological sorting—The process of arranging the vertices in a directed graph without cycles into a topological order.

traversal—An operation that visits (processes) each entry in a data collection.

tree—A data collection that provides a hierarchical, or nonlinear, arrangement of its data. See also *child of a node*, *game tree*, *node*, *parent*, *parse tree*, *root*, and *search tree*.

`try` block—A group of statements that might throw an exception and are contained in braces preceded by the keyword `try`. A `try` block must be followed by one or more `catch` blocks that react to the exception. See also `finally` *block*.

type parameter—Another term for *generic data type*.

U

UML—See *Unified Modeling Language*.

unchecked exception—A runtime exception or an error. Such exceptions need not be handled.

undirected graph—A graph that has at most one edge between any two vertices and whose edges do not indicate a direction. See also *directed graph*.

Unicode—A standard encoding scheme, used by Java, that represents each character by a unique integer regardless of language or platform. See also *ASCII*.

Unicode Transformation Format (UTF)—A character encoding scheme that specifies the number of bytes used to store a Unicode value. Several UTF encoding schemes exist. In particular, UTF-8 uses one byte for each ASCII character, which is sufficient for the characters in the English language. For other characters, UTF-8 uses two, three, or four bytes each. The method `writeUTF` uses a slight variation of UTF-8, known as modified UTF-8. Thus, the method `writeUTF` can represent all Unicode characters, but it saves file space when ASCII characters are written.

UTF-16 uses either two or four bytes to represent a character. Java uses UTF-16, but a `char` value occupies only two bytes, and the method `writeChar` writes two bytes to a file. For strings and `char` arrays, characters requiring four bytes each are represented as pairs of `char` values. Finally, UTF-32 uses four bytes for each character.

unidirectional—A description of an association between two classes that is pictured in a UML class diagram as an arrow pointing in only one direction.

Unified Modeling Language (UML)—A notation used by class designers to describe classes at various stages of their development.

upper bound—A value that is greater than or equal to all possible values of a given item. See also *lower bound*.

upper bound of a wildcard—Any subtype of a class `C`, as indicated by the expression `? extends C`.

use case—A notation used in software design to represent a collection of related scenarios.

use case diagram—A notation used in software design that shows the relationships or interactions among various use cases and actors.

use modifier—A Java reserved word—`static`, `final`, `abstract`, `synchronized`, or `volatile`— used in certain declarations of classes, data fields, and methods. This book does not discuss `synchronized` and `volatile`.

user—The person who uses a program.

user interface—The provision a program makes for a person to interact with it.

UTF—See *Unicode Transformation Format*.

V

value—The data represented by a variable or constant.

vector—An object of the standard class `java.util.Vector` that behaves like a high-level array.

vertex—A node in a graph.

visibility modifier—See *access modifier*.

visit—The act of processing a data item during a traversal of a data collection.

W

weight—A value on an edge of a weighed graph. Also known as a *cost*.

weighted graph—A graph whose edges have values, which are known as weights or costs.

weight of an edge—The numeric label on an edge in a weighted graph.

weight of a path—The sum of the weights of the edges in a path in a weighted graph. Also called the *cost of a path*.

white space—Visually empty space in a Java statement or line of output caused by blank characters (spaces), tabs, and new-line characters.

wildcard—The notation `?`, which represents any class type and can be used when defining a generic data type. See also *lower bound of a wild card* and *upper bound of a wildcard*.

worst-case time—An estimate of the maximum time an algorithm requires to solve a problem. See also *average-case time* and *best-case time*.