Appendix A Glossary

Most scientific disciplines have a specialized vocabulary, allowing experts in the field to communicate precisely with each other. Computer science is no different. Gathered here are all the specialized terms used in the text, together with brief definitions. Come here for a quick reminder of a term's definition.

Key Terms

absolute path—A sequence of directories separated by a special character and beginning with a known location that is used to specify the location of a file. *See also* relative path.

abstract class—A class that declares or inherits an abstract method. Such classes are declared using the `abstract` keyword and are often used to declare types in polymorphic programs.

abstract method—A method that does not have a body. Such methods must be declared with the `abstract` keyword.

Abstract Windowing Toolkit (AWT)—A collection of classes used to implement graphical user interfaces.

abstraction—A method of dealing with complexity that eliminates or hides irrelevant details and groups other sets of details into coherent, higher-level chunks.

access modifier—The keywords `public`, `private`, and `protected`. An access modifier controls which clients may have access to the method it modifies.

accessor method—A method that returns the value of an instance variable.

address—A numeric identifier for a particular memory location.

algorithm—A finite set of step-by-step instructions that specify a process.

alias—An alternate name or reference for an object. All of an object's aliases can be used to access the object.

and—A logical connector written in Java as `&&`. The result is `true` if and only if both operands are `true`.

anonymous class—A class without a name. It is used to declare and instantiate a single object at the point where the object is needed.

API—*See* application programming interface.

application programming interface (API)—The set of publicly available methods by which a program accesses the services offered by a class or package.

architecture—The manner in which the most important classes in a program relate to each other.

argument—A value that is copied to a corresponding parameter when a method or constructor is called.

array—A kind of variable that can store many values, each one associated with an index.

assertion—A test that the programmer believes will always be `true` at a particular point in the code.

assignment statement—A statement that gives a new value to a variable on the left side of an equal sign (=).

attribute—An item of information encapsulated in a software object. *See also* instance variable.

avenue—A road on which robots may travel north or south. *See also* road, street.

AWT—*See* Abstract Window Toolkit.

blank final—An instance variable declared to be `final` but not given an initial value until the constructor is executed.

block—1. The statements contained between a matched pair of curly braces. 2. To wait for user input.

body—The statements controlled by the test in an `if` statement or a looping statement.

Boolean expression—An expression that evaluates to either `true` or `false`.

bottom factoring—To remove statements common to both clauses of an `if-else` statement and place them after the `if-else` statement.

bottom-up design—A design methodology that uses available resources to design a solution to a problem. *See also* top-down design.

bottom-up implementation—Implementing a program beginning with methods that perform relatively simple tasks and using them to build more complex methods. *See also* top-down implementation.

bounding box—The smallest rectangle that will enclose a shape drawn on a screen.

breakpoint—An identified place in the source code for a debugger to temporarily stop execution.

buffering—Collecting information until it can all be dealt with at once. Commonly used to improve performance in input and output operations.

bug—A defect in a program.

byte code—An encoding of a program that is more easily executed by a computer than source code. A Java compiler translates source code into byte code.

byte stream—An input or output stream that carries information encoded in binary and is generally not human readable. *See also* character stream, stream.

cascading-`if`—A sequence of `if-else` statements formatted to emphasize that at most, one of several clauses will be executed.

cast—Explicitly converting a value from one primitive type to another compatible type. Also used to assign an object reference to a variable with a more specific type.

character stream—An input or output stream that carries information encoded as characters and is generally human readable. *See also* byte stream, stream.

checked exception—An exception that is checked by the compiler to verify that it is either caught with a `try-catch` statement or declared to be thrown with a `throws` clause. *See also* exception, unchecked exception.

class—The source code that defines one or more objects that offer the same services and have the same attributes (but not necessarily the same attribute values).

class diagram—A graphical representation of one or more classes that show their attributes, services, and relationships with other classes.

class variable—A variable that is shared by all instances of a class. Also called a static variable. *See also* instance variable, parameter variable, temporary variable.

classpath—A list of one or more file paths where the Java system looks for the compiled classes used in a program.

client—An object that uses the services of another object, called the server.

close—To indicate that a program is finished using a file so that resources can be released.

closed for modification—The idea that a mature class should be extended rather than modified when changes or enhancements are needed. *See also* open for extension.

cohesion—The extent to which each class models a single, well-defined abstraction and each method implements a single, well-defined task.

collaborator—A class that works with another class to accomplish some task.

color chooser—A graphical user interface component designed to help a user choose a color.

column-major order—A 2D array algorithm that accesses the array such that the column changes more slowly than the row. *See also* row-major order.

command—A service that changes the state of an object or otherwise carries out some action. *See also* query, service.

command interpreter—A program that repeatedly accepts a textual command from a user and then interprets or executes the command.

comment—An annotation in the source code intended for human readers. Comments do not affect the execution of the program.

comment out code—To put code inside comments so that it is no longer executed when the program is run.

comparison operator—The operators used to compare the magnitude of two values: `<`, `<=`, `==`, `!=`, `>=`, and `>`.

compile—To translate source code into a format more easily executed by a computer, such as byte code.

compiler—A computer program that compiles or translates source code into a format more easily executed by a computer.

compile-time error—A programming error that is found when the program is compiled. *See also* intent error, run-time error.

component—An object such as a button or text box that is designed to be used as part of a graphical user interface.

composition—A relationship between two classes in which one holds a reference to the other in an instance variable. *Also known as* has-a.

concatenation—Joining two strings to form a new string.

concept map—A diagram that uses labeled arrows to connect concepts, represented by a few words.

concrete class—A class that implements the abstract methods named in its superclasses or the methods named in an interface. *See also* abstract class.

console—A window used by a program to communicate with a person using printed characters on lines that appear one after another.

constant—A meaningful name given to a value that does not change.

constraint—An object limiting how a user interface component may be positioned.

constructor—A service provided by a class to construct or instantiate objects belonging to that class.

content pane—The part of a frame designed to display the components of a user interface.

contract—An agreement specifying what client and server objects can each expect from each other.

control characters—Character codes used to control a terminal or printer. Examples include the newline and tab characters.

controller—The part of the Model-View-Controller pattern responsible for gathering input from the user and using it to modify the model. *See also* model, view.

correct—A description of a program that meets its specification.

count-down loop—A loop controlled by a counter variable that is decremented until it reaches zero.

coupling—The extent to which the interactions between classes are minimized.

CRC card—A piece of paper recording the class name, responsibilities, and collaborators for one class during a program's walk-through. CRC is an abbreviation for Classes, Responsibilities, and Collaborators. *See also* walk-through.

cursor—A marker that divides a program's input into the part that has already been read and the part that has not yet been read. Also used to refer to an insertion point.

dangling `else`—A combination of `if` statements and an `else`-clause where it is unclear to which `if` statement the `else` clause belongs.

data acquisition methods—Methods used to obtain data from an input stream, such as the `Scanner` class. *See also* data availability methods.

data availability methods—Queries used to detect the kind of data available to be read from an input stream, such as the `Scanner` class. *See also* data acquisition methods.

debug—The process of removing bugs from a program.

debugger—A tool used to help debug programs by stopping the program's execution at designated points, executing the program one statement at a time, and showing the values of the program's variables.

declaration statement—A statement that introduces a new variable into a program.

deep copy—A copy of an object that also copies any objects to which it refers. *See also* shallow copy.

delimiter—A value such as a space or colon that separates other values or groups of values.

design by contract—A method for designing a program by consistently specifying the pre-conditions and postconditions of each method and invariants on classes as a whole.

detail—An informal term referring to an instance variable or a method.

development cycle—Steps that are repeated while implementing a program, including choosing scenarios, writing code to implement them, testing the result with users, and possibly updating the program's design. One part of a larger development process. *See also* development process.

development process—A process to direct the design and implementation of a program.

documentation comment—A comment designed to be extracted from the source code and used as reference material.

easy to learn—One of five criteria used to evaluate user interfaces. In particular, how well the program supports users learning to use it as well as those deepening their understanding of it. *See also* five Es.

effective—One of five criteria used to evaluate user interfaces. In particular, the completeness and accuracy with which users achieve their goals for using the program. *See also* five Es.

efficient—1. One of five criteria used to evaluate user interfaces. In particular, the speed and accuracy with which users complete tasks while using the program. *See also* five Es. 2. Solving a problem without wasting such resources as memory or time.

element—One item in a collection.

`else` clause—The part of an `if` statement that is executed if the Boolean expression is `false`.

encapsulation—Containing an object's attributes within itself, allowing access to them only via the object's public services.

engaging—One of five criteria used to evaluate user interfaces. In particular, the degree to which the program is pleasant or satisfying to use. *See also* five Es.

enumerated type—A reference type that has a programmer-defined set of values.

enumeration—*See* enumerated type.

equivalence—*See* object equality.

error tolerant—One of five criteria used to evaluate user interfaces. In particular, the degree to which the program prevents errors and facilitates recovery from those that do occur. *See also* five Es.

escape sequence—An alternative means of writing characters that are normally used for another purpose. For example, `\"` to include a double quote in a string.

evaluate—The process of calculating the value of an expression.

evaluation diagram—A diagram showing how an expression is evaluated.

event—An action in the user interface to which the program must respond.

event object—An object containing information about one event.

exception—A type of error message that includes information about how the program arrived at the point at which the error occurred. *See also* checked exception, unchecked exception.

exponent—The part of a number expressed in scientific notation that indicates how far and in which direction the decimal point should be shifted. *See also* mantissa, scientific notation.

expression—A combination of operators and operands that can be evaluated to produce a single value.

extend—To create a new class based on an existing class.

extension—The part of a file's name following the last period and used to indicate the kind of information contained in the file.

factory method—A method that creates and returns an object. Sometimes used as an alternative to a constructor.

field—One item of identifiable information in a record. *See also* record.

file—A place, usually on a disk drive, where information is stored.

file format—The design for how information is organized in a particular file.

final situation—A description of the desired state of a city and all that it contains, including robots, when a program ends. *See also* initial situation.

five Es—Five criteria used to evaluate the quality of user interfaces. *See also* easy to learn, effective, efficient, engaging, error tolerant.

floating point—A computer's internal representation of a number with a decimal point.

flow of control—One sequence of statements, each of which executes completely before the next begins. A program may have several flows of control, each of which is called a thread. *See also* thread.

flowchart—A diagram illustrating the different paths a program may take through a code fragment.

foreach—A variety of `for` loop that accesses each member of a collection one at a time.

format specifier—A code embedded in a format string specifying how one particular value should be formatted when output. *See also* format string.

format string—A string containing one or more format specifiers used to format output.

frame—A window appearing on a computer screen.

garbage—An object that does not have variables referencing it and therefore cannot be used.

garbage collection—The process of removing objects that can no longer be used by a program. *See also* garbage.

graphical user interface—A user interface with a visual representation that sends events to a program. The events are generated via user interaction with input devices such as a mouse or keyboard, and components drawn on the screen such as buttons or text boxes. *See also* event.

hang—A program behaving abnormally such that it does not respond to input and does not complete its execution.

has-a—*See* composition.

hashing—A technique for storing elements of a collection based on a hashcode. Used by some collection classes, such as `HashMap`.

helper method—A method that exists primarily to simplify another method.

high-fidelity prototype—A preliminary version of a program used for evaluation that may perform many of the functions expected in the final program. *See also* low-fidelity prototype, prototype.

host name—The name of a computer connected to the Internet.

identifier—A name for a part of a program, such as a class, a variable, or a method.
immutable—Immutable objects cannot be changed after they are created. *See also* mutable.
implicit parameter—A reference to the object used to call a method. May be accessed within the method with the keyword `this`.
index—The position of one element within an ordered collection, such as an array, a string, or an `ArrayList`.
infinite loop—A loop that lacks a way to affect the termination condition, resulting in its indefinite execution.
infinite recursion—A situation in which a method calls itself repeatedly with no provision for avoiding another call to itself.
information hiding—Hiding and protecting the details of a classes' operation from others.
inherit—To receive capabilities from another class because of a superclass-subclass relationship. The relationship between the classes is sometimes described with the term "is-a." *See also* extend.
inheritance hierarchy—The relationship of several classes that inherit from a common superclass. *See also* inherit.
initial value—The first value a variable is assigned.
initial situation—A description of the desired state of a city and all that it contains, including robots, when a program begins. *See also* final situation.
inner class—A class definition that is contained within the definition of another class, allowing it to access the outer classes' private methods and instance variables.
input—Information that is obtained from outside the program—for example, from the person running the program.
input stream—A stream that carries information from a source to a program. *See also* output stream, source, stream.
insertion point—The point on the console or in a user interface component where the next character typed by the user will appear.
instance—Each object is one instance of a class.
instance variable—A variable that is specific to an object. *See also* class variable, parameter variable, temporary variable.
instantiate—The act of constructing an instance of a class—that is, creating an object.
integer division—Division of an integer by another integer where any remainder or fractional part in the answer is discarded.
intent error—An error in which the program does not produce the desired results, even though it compiles correctly and does not generate run-time errors. *See also* compile-time error, run-time error.
interaction—An informal term referring to a method calling another method or using an instance variable. *See also* detail.
interface—A Java construct listing a set of methods. It is used to define a new type and also to specify that a class belongs to that type because it implements all of the methods listed by the interface.
invoke—To cause an object to perform a specific service.
I/O—An abbreviation for input and output.

IP address—The address of a computer on the Internet.

is-a—*See* inherit.

java archive (jar) file—A single file containing many compiled classes, making the classes easier to distribute.

javadoc comment—*See* documentation comment.

key—A value used to uniquely identify another value.

keyboard focus—A property of at most one component in a user interface, the component that will receive input from the keyboard.

keyword—The words defined by the language to have special meaning and that cannot be used as identifiers. Examples include `class`, `while`, and `int`. Also called a reserved word.

layout—The act of arranging components in a graphical user interface.

layout managers—An object that manages the layout of components in a graphical user interface.

left justified—Elements (typically lines of text) that are aligned vertically on the left side. *See also* right justified.

lexicographic order—Ordering strings by comparing their individual characters.

library—A collection of resources available to be used in many different programs. *See also* package.

lifeline—A part of a sequence diagram that shows the lifetime of an object.

lifetime—The time in which values are preserved in a variable before they are destroyed by either the object containing them being garbage-collected or the variable going out of scope.

list—An ordered collection of elements, perhaps with duplicates. *See also* map, set.

listener—An object registered with a component that responds to events generated by the component.

local variable—*See* temporary variable.

logic error—*See* intent error.

logical negation operator—The operator `!`. It negates the Boolean expression following it. *See also* negate.

loop—A statement that repeats the statements it controls. A `while` statement is a form of loop.

loop-and-a-half—A loop that must execute part of its body one more time than the rest of the body. Typically implemented with duplicate code before or after a `while` loop or with a `while-true` loop.

low-fidelity prototype—A model of a program used for evaluation purposes that only approximates the final design, perhaps using paper and pencil. *See also* high-fidelity prototype, prototype.

mantissa—The fractional portion of a number expressed in scientific notation. *See also* exponent, scientific notation.

map—An object storing a collection of objects, each identified by a key. *See also* key, list, set.

memory—Part of the computer hardware that stores information, such as variables and program instructions.

message—A client object sends a message to a server object to invoke one of its services.

method—The source code that implements a specific service.

method resolution—The process of determining the correct method to execute in response to a method call.

mixin—A type, defined by an interface, that supplements the primary type of a class.

model—1. A simplified description of a problem, usually in a formal notation such as mathematics or a computer program, that enables people to forecast the future, make decisions, or otherwise solve the problem. 2. The part of the Model-View-Controller pattern that models or abstracts a problem. 3. To create a simplified description of something to help us make decisions, predict future events, or maintain relevant information.

multi-line comment—A comment that may span multiple lines. It begins with `/*` and ends with `*/`. *See also* comment, documentation comment.

multiplicity—The notation on arrows in a class diagram indicating how many instances of a class are used by an object.

mutable—A mutable object can be changed after it has been created. *See also* immutable.

natural language—Language used in everyday speech.

negate—To make a Boolean expression return the opposite value.

nest—To place a control statement such as `if` or `while` within another control statement.

nested loop—A loop that occurs within another loop.

newline character—A character that divides two lines of text. It can be represented in a string with the character sequence '`\n`'.

null—A special value that can be assigned to any object reference, meaning it does not refer to any object.

object diagram—A diagram that shows one or more specific objects and the values of their attributes.

object equality—Tested with the `equals` method. Establishes whether two object references refer to objects that are equivalent. *See also* object identity.

object identity—Tested with ==. Establishes whether two object references refer to the same object. *See also* object equality.

object-oriented programming language—A computer programming language incorporating the ideas of encapsulation, inheritance, and polymorphism.

open—Preparing a file for input or output.

open for extension—A class that is written in such a way that it can be modified through inheritance. *See also* extend, inherit.

operand—The value, variable, or query on which an operation is to be done. *See also* operator.

operator—A symbol denoting an operation, such as addition or division, to be performed on its operands. *See also* operand.

or—A logical connector written in Java as `||`. The result is `true` if and only if at least one of the operands is `true`.

origin—The place from which measurement begins. In a robot city, the intersection of street 0 and avenue 0. On a computer screen, the upper-left corner.

output—Information that is produced by a program and displayed on a screen or written to a file.

output stream—A stream that carries information from a program to a sink or destination. *See also* input stream, sink, stream.

overload—Two or more methods with the same name but different signatures are overloaded. The Java system chooses which one to execute based on the actual parameters used when the method is called. *See also* signature.

override—Replacing a method in the class being extended with a new version of the method.

package—A group of classes, usually organized around a common purpose.

parameter variable—A type of variable used to communicate a value to a constructor or service to use in accomplishing its purpose. *See also* class variable, instance variable, temporary variable.

partially filled array—An array that uses the elements with indices 0..*n*-1 to store values, where $n \leq s$, the size of the array. *n* is stored in an auxiliary variable.

picture element—A small dot displayed on a computer screen. Many picture elements compose the image displayed. Often abbreviated as "pixel."

pixel—*See* picture element.

point—A unit of measurement, used for fonts, equal to 1/72 of an inch.

polymorphism—Setting up two or more classes so that objects can be sent the same message but respond to the message differently—that is, in ways appropriate to the kind of object receiving the message.

postcondition—A statement of what should be true after a method executes. *See also* precondition.

precedence—A rule that determines which operations are done first when an expression is evaluated.

precision—The closeness of the approximation between a value stored in the computer and the actual value.

precondition—A situation that must be true when a method is called to ensure that it executes correctly. *See also* postcondition.

predicate—1. A query (method) that returns a value of either `true` or `false`. 2. The part of a sentence that contains a verb and explains the action or the condition of the subject. *See also* subject.

primary key—When sorting records, the primary key is the most important determinant of the order. *See also* secondary key.

primitive—The most basic available methods out of which more complex methods are built.

primitive type—A type whose values can be manipulated directly by the underlying hardware. In Java, they include `int`, `double`, and `boolean`. *See also* reference type.

processing stream—A stream that processes information as it flows from a source to a sink. *See also* provider stream, stream.

program—A detailed set of computer instructions designed to solve a problem.

prompt—An indication to the user that some action is required. A prompt is usually printed on the screen just before input is required from the user.

prototype—A preliminary version of a program used for evaluation or learning purposes. *See also* high-fidelity prototype, low-fidelity prototype.

provider stream—A stream that provides information from a source or to a sink. *See also* processing stream, sink, source, stream.

pseudocode—A blend of a natural language and a programming language, allowing people to think more rigorously about programs without worrying about programming language details.

query—A service or method that answers a question. *See also* command, method, service.

random access—A property of an information collection where every item can be accessed as easily and as fast as every other item.

range—The number of different values belonging to a type such as `int` or `double`.

read—Obtaining input from a file or other input stream.

record—A collection of information pertaining to one thing (for example, an employee) in a file that typically contains information about many of those things. *See also* field.

refactor—The process of modifying a program to improve its overall quality without changing its functionality.

reference—The information stored in a variable that refers or leads to a specific object.

reference type—A type whose values are defined by a class or an interface. *See also* primitive type.

reference variable—A variable that refers to an object or contains `null`.

register—Adding an object to a list of objects that should be notified when certain events occur.

relative path—A sequence of directories that gives a file location relative to the current working directory. *See also* absolute path, working directory.

reliability—A characteristic of quality programs in which the program does not crash, lose, or corrupt data, and is consistent in how it operates.

remainder operator—An operator that returns the remainder or part that is left after dividing one integer by another. *See also* integer division.

requirements—A written statement of what a program is supposed to do. *Also called* specifications.

reserved word—*See* keyword.

responsibility—The things a class must do to support the operation of the program. Identified during the design of the program.

return—The action of going back to the statement that called the currently executing method. If the method is a query, it also provides a value to the expression from which it was called.

return type—The type of the value returned by a query. Specified just before the method's name when it is declared.

right justified—Elements (typically lines of text) that are aligned vertically on the right side. *See also* left-justified.

road—A street or an avenue on which a robot may move between intersections. *See also* avenue, street.

row-major order—A 2D array algorithm that accesses the array such that the row changes more slowly than the column. *See also* column-major order.

run-time error—An error detected when a program executes or runs because it has executed an instruction in an illegal context. *See also* compile-time error, intent error.

scenario—A specific task that a user may want to perform with the program. *Also known as* use case.

scientific notation—A number expressed as the multiplication of a fractional number (the mantissa) and 10 raised to some power (the exponent). *See also* exponent, mantissa.

scope—That part of a program where an identifier is available for use.

search—The process of attempting to locate one value in a collection of values.

secondary key—When sorting records, the secondary key is used to determine the order of records that have equal primary keys. *See also* primary key.

self-documenting code—Code that is written to minimize the need for documentation. Well-chosen identifiers are the key tool used in writing self-documenting code.

semantics—The meaning of a statement. *See also* syntax.

sequence diagram—A diagram showing the sequence of activities among cooperating objects.

serif—Short lines at the end of each stroke of a printed letter.

server—An object that provides services to a client object. *See also* client.

service—An action that an object performs in response to a message. Services are subdivided into queries and commands. *See also* command, message, method, query.

set—An unordered collection of unique objects. *See also* list, map.

shallow copy—A copy of an object that does not copy any objects it references. *See also* deep copy.

short-circuit evaluation—Evaluating a Boolean expression so that sub-expressions that cannot affect the result are not evaluated.

side effect—A change in state caused by executing a method.

signature—The name of a method, together with an ordered list of all the types of its parameters.

simulate—*See* trace.

single-line comment—A comment extending from a double slash (`//`) until the end of the line. *See also* multiline comment, documentation comment.

sink—The destination for information flowing in a stream. *See also* source, stream.

software object—An abstraction in an object-oriented program used to model a real-world entity.

source—The origin of information that flows in a stream. *See also* sink, stream.

source code—The words and symbols written by programmers to instruct a computer what to do.

spaghetti code—A derisive description of source code written with undisciplined use of a `goto` construct (which Java does not have). *See also* structured programming.

special symbols—Symbols that have a special meaning in the Java language, including braces, parentheses, and the period and semicolon characters.

specification—*See* requirements.

stack trace—An ordered list of which methods called which methods, extending from the point an exception is thrown back to the `main` method.

state—The state of being of an object as defined by the contents of its attributes.

state change diagram—A diagram that shows how an object's state changes over time.

statement—An individual instruction in a programming language.

static variable—*See* class variable.

stepwise refinement—A method of writing programs where each method is defined in terms of helper methods, each of which implement one logical step in solving the problem. *Also known as* top-down design.

Strategy pattern—A pattern where one object uses another object that defines one algorithm from a family of algorithms, making it easy to change the behavior of the first object.

stream—An ordered collection of information that moves from a source to a destination or sink. *See also* byte stream, character stream, input stream, output stream, processing stream, provider stream.

street—A road on which robots may travel east or west. *See also* avenue, road.

structured programming—A programming discipline that restricts how flow of control can be shifted from one part of the program to another. *See also* spaghetti code.

stub—A method that has just enough code to compile, but not enough to actually do its job.

subclass—A class that receives part of its functionality from a superclass. *See also* extend, inherit, superclass.

subject—The part of a sentence that says who or what did the action. *See also* predicate.

substitution principle—A key principle underlying polymorphism where an object of one type, *A*, can substitute for an object of another type, *B*, if *A* can be used anyplace that *B* can be used. *See also* polymorphism.

superclass—A class that has been extended to create a subclass. *See also* extend, inherit, subclass.

Swing—A newer addition to the collection of classes available to write graphical user interfaces in Java. *See also* Abstract Window Toolkit.

syntax—The form of a statement. *See also* semantics.

tab stop—A predefined location where the insertion point will be located after a tab is inserted into text.

tag—A keyword such as `@param` or `@author` used to identify standardized information in documentation comments. *See also* documentation comment.

template method—A method implementing the common part of a problem that has several variations. The differences between the variations are expressed in helper methods contained in subclasses.

temporary variable—A variable defined within a method. The variable and the information it contains are discarded when the method finishes execution. *See also* class variable, instance variable, parameter variable.

test harness—A program used to test a method or class.

`then` clause—The statements that are executed when the test in an `if` statement is `true`.

thread—A sequence of statements that executes independently of other sequences of statements. The execution of two or more threads may be interleaved. *See also* flow of control.

throw—The action of interrupting the normal execution of a program with an exception.

token—A group of characters separated by delimiters. *See also* delimiter.

top factor—The process of removing redundant statements from the beginning of both clauses in an `if` statement.

top-down design—Designing a program or a method by dividing it into logical pieces that work together. These pieces are themselves designed using top-down design. This process continues until a piece is so simple that it can be solved without dividing it. *See also* stepwise refinement, top-down implementation.

top-down implementation—Implementing a method by writing it in terms of helper methods. Helper methods may also be defined in terms of other helper methods. Eventually, each helper method will be simple enough to implement using existing methods or without using helper methods. *See also* bottom-up implementation, top-down design.

trace—To execute a program without the aid of a computer, usually by recording state changes in a table. Also called simulate.

type—A designation of the valid values for a variable or parameter.

unchecked exception—An exception that the compiler does not require to be caught with a `try-catch` statement or declared to be thrown with a `throws` clause. Used for errors from which recovery should generally not be attempted. *See also* checked exception, exception.

Unicode—A character encoding standard that allows up to 65,536 different characters to be defined.

usability—A criteria of a program's quality from a user's perspective, determined by the effort required to learn, operate, prepare input, and interpret output when compared to the alternatives.

use case—*See* scenario.

validation—Determining if the intent of a program or program fragment is correct. *See also* verification.

value—One item of information stored in a map collection that is identified by a key. *See also* map.

variable—A named place where a program can store information. *See also* instance variable, parameter variable, temporary variable.

variable declaration—A programming language statement that introduces a variable in the source code and specifies its type.

verification—Determining if a program or program fragment correctly implements the intended functionality. *See also* validation.

view—The part of the Model-View-Controller pattern that displays relevant information from the model to the user. *See also* controller, model.

walk-through—The process of simulating the execution of a program using other people, each of which takes on the role of one or more classes.

wall—An element of a robot's environment that it cannot move through.

waterfall model—A development process in which the output of one phase is the input to another phase. The waterfall model does not explicitly include iteration. *See also* development process.

whitespace—Characters such as spaces and tabs that appear as white space when printed on paper.

working directory—An executing program's default directory. Files are read and written in the working directory unless their name includes an absolute or relative path.

wrapper class—A class whose only variable is a primitive, such as `int` or `double`. Such classes exist so that a primitive value can be treated as an object.

write—The process of placing information in a file. *See also* read.