Introduction to Interactive Programming
by Lynn Andrea Stein
A Rethinking CS101 Project

Glossary

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A

abstract method
: A method with no body; a method signature followed by a semicolon.

abstraction
: The use of a general contract as a placeholder for specific details. A technique for separating use from implementation.

alternative
: In a conditional statement, the optional sub-statement to be executed if the boolean test expression's value is false.

ampersand
: & Used in conjunction.

applet
: A Java program capable of running embedded in a web browser or other specialized applet environment. Contrast application.

application
: A Java program capable of running "standalone". Contrast applet.

animacy
: A Java Thread that enables concurrent execution, e.g., of a self-animating object. See the chapter on Self-Animating Objects.

animate object
: See self-animating object.

array
: A structure for holding many Things of the same type.

argument
: A value supplied to a method when it is invoked. During the execution of the method body, this value is named by the matching method parameter.

arithmetic operator
: An operator that computes one of the arithmetic functions. See the chapter on Expressions.

assertion
: A statement of what must be true. In Java, assertions are generally found in comments.

assignment
: The association of a name with a value. See the chapter on Things, Types, and Names.

: Also the operator in such an assignment. See the chapter on Expressions.

assumption
: Something taken for granted by a computer program or its designer. Often a requirement on the environment in which a program operates. When an assumption is violated, the program may not behave properly, so it is especially important for a software engineer to document the assumptions of the program.

asterisk
: * Sometimes called star. Used as the multiplication operator and, together with a slash, to delineate certain comments.

B

backslash
: \ Used in character escapes.

bang
: See exclamation point. Also a loud noise, often made by a child.

base type
: In an array, what (type) it is an array *of*.

base case
: In a recursive definition, the case that does not rely on the thing being defined. Contrast recursive case.

batch
: Things that happen while you wait....and wait....and wait. Or, better yet, things that you leave behind and pick up later. Contrast real time.

binary operator
: An operator that takes two operands. See the chapter on Expressions.

binding
: See name binding.

bit
: A single binary digit.

bitwise operator
: An operator that computes a bit-by-bit function such as bitwise complement. See the chapter on Expressions.

block
: A sequence of statements contained between braces. See the section on Blocks in the chapter on Statements.

blocking
: Waiting for information to become available, especially in a read.

body
: The body of a method, class, or interface, i.e., either a method body, a class body, or an interface body.

boolean
: A true-or-false value. In Java, represented by the primitive type `boolean` and by the object type `Boolean`. See the sidebar on Java Primitive Types in the chapter on Things, Types, and Names.

boolean expression
: An expression whose type is `boolean`.

boot, boot up
: Start up (a computer or program).

bottom up, bottom-up design
: An approach to design that starts by identifying the simplest, most concrete things in your system and proceeds by combining them. Contrast top down.

brace
: { or } Used to enclose bodies or blocks.

bracket
: [or] Used in array expressions.

bug
: An error in a program. Contrast feature.

C

call
: See invocation.

call path
: The sequence of method invocation (instructions followed by a Thread) that led up to the currently executing method body. Unless execution exits abruptly, each of these invocations will return, one at a time, in this order, along the reverse of the call path, i.e., the return path.

carriage return
: One of two line-ending characters. (The other is line feed.) So named after an archaic device called a typewriter in whose early models the carriage (i.e., paper-bearing part) literally needed to be returned to the other side of the typewriter at the end of each line.

case-sensitive
: Distinguishing between upper and lower case letters.

cast expression
: An expression involving a type and an operand whose value is the same as its operand but whose type is the type supplied. Contrast coercion.

catastrophic failure
: An exceptional circumstance so incapacitating that your program cannot hope to prevent or deal with it. At this point, the only hope is in recovery.

`catch` statement
: A particular kind of Java statement, typically used with exceptions, that receives a thrown object. See the chapter on Exceptions.

character
: A single letter, digit, piece of punctuation, or piece of white space. In Java, represented by the primitive type `char`, using Unicode notation, and occupying sixteen bits, and by the object type `Char`. See the sidebar on Java Primitive Types in the chapter on Things, Types, and Names.

character escape
: A special sequence indicating a character other than by typing it directly. Especially useful for non-printing characters, such as carriage return.

child
: Offspring, inheritee, extender. The opposite of a parent.

class

A (user-definable) type from which new objects can be made. See the chapter on Classes and Objects.

class body
: The portion of a class definition containing the class's members. The portion of a class definition enclosed by { }. See the chapter on Classes and Objects and the Java Chart on Classes.

class object
: The object representing the class itself, i.e., the factory. Itself an instance of class `java.lang.Class`.

client
: With respect to some service, the (computational) entity that needs that service. Contrast server.

client pull
: A communication pattern in which the client initiates the service. Contrast server push.

code
: An excerpt from a program. Formally, source code is compiled (typically into executable code).

coercion
: Treating an object of one type as though it were of another type. Contrast cast. See the chapter on Expressions.

colon
: : Used in the ternary conditional expression and after labels.

comment
: Text embedded in a program in such a way that the Java compiler ignores it. Intended to make it easier for people to read and understand the code.

comparator
: An operator in an expression of `boolean` type.

completeness
: A promise made by a system that it will supply all true (or relevant) information. Trivially (and not very interestingly) accomplished by providing all information, whether true or not. Contrast soundness.

component
: A member, especially a field.

compiler
: The utility that transforms your Java code into something that can be run on a Java virtual machine.

compile time
: The time at which a program is compiled. Compile time information refers to information that is available by reading the (partial) source code of the program. Contrast run time, information available when the program is actually executing.

compound assignment
: A shorthand assignment operator (or expression) that also involves an arithmetic or logical operation.

concatenation
: The gluing together of two strings.

condition

In a conditional statement, the boolean expression whose value governs whether the consequent or the alternative is executed.

conditional
: A compound statement whose execution depends on the evaluation of a boolean expression. Consists of a condition, a consequent, and an optional alternative. In Java, often an `if` statement. Contrast sequence, loop.

conjunction
: The logical operator && (and).

concurrent
: Literally or conceptually at the same time. Contrast sequential.

consequent
: In a conditional statement, the sub-statement to be executed if the boolean test expression's value is true.

console
: See Java console.

constant
: A name associated with an unchanging value. Typically declared `final`.

constant expression
: Any expression whose value can be determined at compile time, i.e., independent of any execution of any Threads of any program. Typically either a literal or a name declared `final` .

constructor
: The code which specifies how to make an instance of a class. Its name matches the name of the class. A constructor is a class member. See the chapter on Classes and Objects.

controller
: (In a GUI.) How the view is connected to the model. In java.awt, this is not usually a separate object.

D

data
: Values, as opposed to executable code. Things that might be associated with names such as variables, parameters, or fields. See also state.

data repository
: A kind of object whose primary purpose is to store data. See the chapter on Designing with Objects.

debug
: To attempt to eliminate bugs from your program.

declaration
: A statement associating a name with a type. Once the name has been declared, it can be used to refer to Things of the associated type. See the chapter on Things, Types, and Names.

declarative programming language
: A programming language based on declarations and assertions, i.e., statements, generally containing variables, about what must be true. A declarative programming language has rules of execution that calculate values for which these statements hold. All computation in a declarative programming language is implicit. Contrast functional, imperative, and object oriented programming languages.

default value

The value associated with a name that has been declared but not assigned a(n initial) value. See the sidebar on Default Initialization in the chapter on Things, Types, and Names.

default visibility
: Also called package visibility. The visibility level of an unmodified interface, class, method, field, or constructor. Visible to only within the package.

definition
: A statement that both declares and initializes a name. See the chapter on Things, Types, and Names.

delegation
: See event delegation.

design
: The process of figuring out what your program should do and how it should accomplish it.

design for modifiability
: The design principle that says that you should build software that is easy to maintain and adapt. See software lifecycle.

designer
: A software engineer while working on the design of a program.

dial name
: A name capable of referring to something of a primitive type, whose value is encoded directly in the memory reserved by the name. The types named by dial names are formally called value types. See the chapter on Things, Types, and Names.

disk server
: A computer that provides (access to) disk storage for other computers on a network.

dispatch
: A control-flow management technique in which you decide how to respond by considering the value that you have been asked to respond to (as opposed, e.g., to other environmental factors).

dispatch on case

A situation in which the decision of what action to take depends on which of a set of known values matches the value of a particular expression. A special case of dispatch. In Java, often implemented with a `switch` statement.

disjunction
: The logical operator `||` (or).

dot
: See period.

double precision floating point
: A representation for rational numbers (and an approximation for real numbers) that uses 64 bits of storage. In Java, implemented by the primitive type `double`. See floating point.

down cast
: A cast from superclass to subclass. May be invalid; should be guarded.

E

embedded

The property of being in an environment (or system) and interacting with it.

encapsulation
: Packaging up of specific details into a single manipulable unit, often one that hides these (implementation) details from the user.

entity
: A member of the community. A conceptual unit consisting of an object or set of objects that is (implicitly or explicitly) persistent and that interacts with other entities.

environment

: Where an entity is embedded. What the entity interacts with.

error checking
: Code (often a conditional statement) designed to catch illegal values or other potential problems, and to correct them, before or as they arise. A way to avoid bugs in your program. An important part of design.

evaluate
: To compute the value of an expression.

event
: 1. Something that happens.

: 2. A special kind of object used in event-driven programming to record the occurrence of a particular event (in the conventional sense). See the chapters on Event-Driven Programming and Event Delegation.

event-driven programming
: A style of programming in which an implicit (often, system-provided) control loop activates event handler methods when a relevant event occurs. See the chapters on Event-Driven Programming and Event Delegation.

event delegation
: The system by which a separate listener object provides event handlers for another (GUI Component) object. Used in Java AWT versions 1.1 and later. See the chapter on Event Delegation.

event handler

: In event-driven programming, a method that is called when a relevant event occurs. See the chapters on Event-Driven Programming and Event Delegation.

exclamation point
: ! Used in boolean negation.

exception
: A special kind of Java object used to indicate an exceptional circumstance. Typically used in conjunction with `throw` and `catch` statements. See the chapter on Exceptions.

execute
: To follow the instructions corresponding to a statement or program.

exit condition
: The condition under which the repeated execution of a loop stops. Formally called the termination condition for the loop.

explicit
: What I'm telling you. Contrast implicit.

expression

A piece of Java code with a type and a value, capable of being evaluated. Contrast statement. See the chapter on Expressions.

extend

To reuse the implementation supplied by a superclass (or, for interfaces, a parent interface) through inheritance.

F

factory

A class, metaphorically, for its instances.

feature

1. A deliberately designed and generally beneficial aspect of a program.

2. *post hoc.* A bug, when discovered by a user after it's too late to fix it.

field

A data member of a class, i.e., a name associated with each instance of a class (if not `static`) or with the class object itself (if `static`). See the chapter on Classes and Objects.

field access

An expression requiring an object and a field name. Its type is the declared type of the field and whose value is the value currently associated with that field.

final

A modifier indicating

1. that the value associated with a name, once assigned, cannot be changed, or

2. that a method cannot be overridden in a subclass, or

3. that a class cannot be extended.

file

A collection of information stored as a single unit on a persistent storage medium such as a computer disk.

file server

A computer that provides (access to) a set of files for other computers on a network.

floating point

A representation for rational numbers (and an approximation for real numbers) that uses 32 bits of storage. In Java, implemented by the primitive type `float`. Contrast double precision floating point.

flow of control

The sequence of instructions executed. Certain statements (such as conditionals and loops) modify the flow of control. Also called control flow.

footprint

See method footprint.

function
: A method, especially one with a return value.

functional programming language
: A programming language based on expressions rather than statements, i.e., in which (almost) everything has a value. Functional programming languages minimize the role of assignment. Contrast object oriented, functional, and imperative programming languages.

functional recursion
: A form of recursion in which a function--or Java method--is defined recursively. See recursive function. Sometimes called procedural recursion, especially when the recursive method has no return value. Contrast structural recursion.

G

getter, getter method
: A method that exists solely to provide read access to a field. Formally called a selector.

global variable
: A term with no meaning in Java.

grandparent
: A parent's parent. Who to call when the parent falls through.

graphical user interface
: A user interface that makes use of windows, icons, mouse, etc., and is typically implemented in an event-driven style. Sometimes abbreviated GUI.

guarantees
: See promises.

guard expression
: A test that prevents execution of a potentially dangerous statement.

GUI
: An acronym for graphical user interface.

H

hyphen
: - Used as the unary and binary subtraction operator and to indicate negative numbers.

I

identifier
: The formal term for a name.

idiot proofing
: A not very tactful name for error checking, especially as concerns interaction with the user.

`if`, `if/else`
: Java's conditional statement.

imperative programming language
: A programming language based on statements rather than on objects. The major units of imperative programming languages are procedures, typically without return values. (These are roughly void

methods, but without encapsulating objects.) Contrast object oriented, imperative, and declarative programming languages.

implement

What an implementor does. More specifically, what a class does with an interface. Also what a programmer does.

implementor

The 1. person or 2. entity that provides the implementation for an interface or contract. Contrast user.

implementation

Executable code. Also "how to". Contrast use.

implicit

What I'm not telling you, but is so anyway. Contrast explicit.

incremental program design

The design-build-test-design cycle in which every attempt is made to keep the program working at all times and to make only minor modifications between tests.

index

An expression, typically with an integer value, used to select a member from a (generally uniform) set.

inherit

What a child does with a parent. Specifically, what a subclass does with a superclass.

inheritance

The process by which one class shares the definition and implementation provided by another. Also the process by which one interface extends another. Uses the Java keyword `extends`. See the chapter on Inheritance.

initialization

The assignment of an initial value to a name or, by extension, to an object's fields.

input

Information that is read by a program or entity; or, the stream or other resource from which input is read.

instance

An object created from a class, whose type is that class. See the chapter on Classes and Objects.

instantiate

To create an instance from a class, typically through the use of a constructor (and `new`).

instruction follower

The thing that executes statements. In Java, a Thread.

instructions

Code, generally statements, explaining how to do something. Followed step by step by an instruction follower.

integer type

In Java, one of `byte`, `short`, `int`, `long`, `char`, or `boolean`. Expressions of these (and only these) types may be used as the test expression of a `switch` statement.

interface
1. The common region of contact between two or more entities.

2. (Java) A formal statement of method signatures and constants defining a type and constraining the behavior of objects implementing that interface.

See the chapter on Interfaces.

interface body
The portion of an interface definition containing the interface's members. The portion of an interface definition enclosed by { }.

interaction
Literally, *action between*. How two (or more) things get along.

invocation
To call a method, i.e., execute its body, passing arguments to be associated with the method's parameters.

J

Java console
A place in every Java environment from which standard input is read and to which standard output is written. I/O to the Java console is provided by `cs101.util.Console`, `java.lang.System.in`, and `java.lang.System.out`.

jelly
An exceedingly sticky concoction made from the juice of a fruit, often a grape, ideally purple. See also peanut butter.

just going around in circles
What happens in a recursion without a base case.

K

keyword
A word with special meaning in Java. All Java keywords are reserved, i.e., cannot be used as Java names.

L

label
A no-op marker statement that simply indicates its location in code for later reference, e.g., by a break or continue statement. Not related to the more common label name.

label name
A name capable of referring to something of an object type, i.e., anything not of a primitive type. See the chapter on Things, Types, and Names.

layered
Especially layered service. A service that is accomplished through reliance on another service.

lector

One who reads.

left-hand side

In an assignment, the expression representing the shoebox or label to which the value is assigned.

legacy

Old, often out-of date. Typical in "legacy software", a piece of software that an organization continues to use (and that a software engineer must therefore adapt, integrate, or interface with) long beyond its desirable lifetime.

literal

A Java expression to be read literally, i.e., at face value. Only the primitive types plus strings have corresponding literal expressions. See the sidebar on Java Primitive Types in the chapter on Things, Types, and Names.

local

Another term for a variable. Short for local variable.

local variable

The formal term for a variable.

logical operator

An operator that computes an arithmetic function such as conjunction or disjunction. See the chapter on Expressions.

loop

A construct by which a sequence of statements is executed repeatedly, typically until some exit condition is met. Contrast sequence, conditional.

lossy

Losing information. For example, a narrowing coercion may be lossy because the thing being coerced may be too big to fit into the new type.

M

magic number

A literal number appearing without explanation or obvious meaning in your code. It is generally better style to use a constant, i.e., a final name.

mail server

A computer that provides (electronic) mail service for other computers on a network.

maintain

To continually test, debug, and modify a program so as to fix bugs and otherwise ensure that it continues to work reliably (e.g., in the face of changes to its environment, to its requirements, or to the underlying system).

member

A constructor, field, or method of a class. Alternately, a (static) field or (abstract) method of an interface. Also member (inner) classes or interfaces. See the chapter on Classes and Objects.

method

An executable class member. Consists of a signature plus a body (unless `abstract`). When a method is invoked on an argument list, the body is executed with each of the method's parameter names bound to its corresponding argument.

method body
: The portion of a method that contains executable statements. When a method is invoked (on a list of arguments), its body is executed within the scope of the parameter bindings, i.e., with the parameter names bound to the corresponding arguments.

method footprint
: The name plus the ordered list of parameter types of a method. An object may have at most one method with any particular footprint. Contrast method signature. See the chapter on Interfaces.

method invocation
: See invocation.

method overriding
: When a subclass redefines a method or field that would otherwise be inherited from its superclass.

method overloading
: When one object has two or more methods with the same name (but different footprints), typically performing different functions.

method signature
: The specification of a method's name, ordered list of parameter types, return type, and exceptions, possibly including modifiers. Contrast method footprint. See the chapter on Interfaces.

model
: (In a GUI.) An object implementing how the mechanism works, i.e., what it does within your program. Contrast view.

modifier
: A formal Java term such as `abstract`, `final`, `public`, `static`, `synchronized`, etc., which is used in the definition of a class, interface, or member. See the Java Charts for details.

mutator
: The formal name for a setter method.

N

name
: A Java expression that refers to a particular object or value. Examples include variables, parameters, fields, class names, and interface names. Every name has an associated type (fixed when the name is declared). Within its scope, the name is generally bound to a value (of the appropriate type). See the chapter on Things, Types, and Names.

name binding
: The association of a name with a value, typically through assignment or through parameter binding during method invocation. The details of this association depend on whether the name is a shoebox name or a label name, i.e., of primitive or object type.

narrowing, narrowing coercion
: Treating a thing of one type as though it were of another, smaller, less precise type. Includes coercion to a smaller primitive type (e.g., long to int) as well as coercion to a superclass (or super-interface) type.

natural language
: What humans speak (before they become geeks).

negation
: Not. Of a boolean, the other one.

network
: A set of interconnections, often between computers.

no-args
: Taking no arguments or, more properly, having no parameters.

no-op
: Having no effect, like talking to a wall or shouting into the wind.

`null`
: A Java keyword. The non-value with which an unbound label name is associated.

null character
: The character with unicode number 0. Not to be confused with the non-value `null`.

O

object
: A non-primitive, non-`null` Java Thing. An instance of (a subclass of) `java.lang.Object`.

object oriented programming language
: A programming language based on objects. Contrast functional, imperative, and declarative programming languages.

object type
: In Java, any type other than one of the eight primitive types. All object types are named by label names.

operand
: One sub-expression of an operator expression. See the chapter on Expressions.

operator
: The part of an operator expression that determines the particular relationship of the operands to the expression's value. See the chapter on Expressions.

operator expression
: An expression involving an operator (e.g., +) and one or more operands. Typically, the value of the expression is a particular function of the operands, with the operator specifying what function. See the chapter on Expressions.

overriding
: See method overriding.

overloading
: See method overloading.

output
: The information that is written by a program or entity; or, the stream or other resource to which it is written.

P

package
: 1. A named group of Java interface and class definitions.

  2. The default visibility level of an unmodified interface, class, method, field, or constructor. Visible only within the package(1).

paper

An archaic but amazingly persistent storage medium made of wood pulp. Reported continually over the last half-century to be destined for imminent obsolescence with the incipient advent of the paperless office. Sometimes used with a typewriter.

parameter
: A name whose scope is a single invocation of the method to which it belongs. Declared in the method signature. When the method is invoked on a list of arguments, each parameter is bound to the corresponding argument prior to (and with scope over) the execution of the method body.

parameter binding
: The form of name binding that occurs when a method is invoked on a list of arguments. Each of the method's parameters is bound to the corresponding argument, i.e., the first parameter to the first argument, etc.

parent, parent type
: A generic term encompassing superclass, interface implemented, or interface extended. Also, the common enemy uniting child and grandparent.

peanut butter
: A gooey brown paste made by grinding up a certain legume, often consumed with jelly between two slices of very bland white bread.

period
: . Sometimes also called dot. Used in method invocation and field access expressions, package naming, and as a decimal point.

persistent
: Existing even when not currently the subject of the coder's, computer's, or instruction-follower's attention.

pipe
: See vertical bar.

pointer
: A term with no meaning in Java.

polymorphism
: Behaving differently with different types of things.

port
: To translate a piece of software from one programming language to another or from one kind of computer system to another.

postcondition
: What is true after something has happened. Typically indicates something that has changed. Contrast precondition.

postfix
: Coming after.

precondition
: What must be true before something can happen. Contrast postcondition.

predicate
: An expression or method whose (return) value is of type boolean.

prefix
: Prior to.

primitive type
: In Java, one of `byte`, `short`, `int`, `long`, `float`, `double`, `char`, or `boolean`. All primitive types are named by shoebox names. See the sidebar on Java Primitive Types in the chapter on Things, Types, and Names.

`private`
: A Java keyword. A class or interface member declared private is visible only within the body of its defining class or interface.

procedural abstraction
: Combining a group of instructions into a single named unit (a procedure; in Java, a method) so that it can be reused.

procedural recursion
: See functional recursion.

procedure
: Something that is done. In Java, a method, especially (but not exclusively) one without a return value.

program
: n. A collection of executable code. The how-to instructions that a computer follows.

: v. To compose a program. See also incremental program design, debug.

programmer
: A person who develops (designs, writes, debugs, modifies) a program.

programming language
: A language in which one writes a program. For the purposes of this book, Java.

programming environment
: A set of on-line tools in which a programmer develops (designs, writes, debugs, modifies) a program. Typically includes (at least) an editor, compiler, runtime environment, and debugger.

promises
: Commitments a program (or its designer) makes about that program's behavior. Some promises are embodied in the program's interface.

`protected`
: A Java keyword. A class or interface member declared protected is visible within its package and within any class (or interface) that extends (or implements) its containing class (or interface).

prototype
: A simple, often hastily thrown together version of a program (or other product) intended to help in the design process. A prototype is not intended for serious use and may lack the features or complexity of the final, production version.

`public`
: A Java keyword. An interface, class, method, field, or constructor declared public is visible everywhere.

Q

query

A specialized request to a program or other entity, in which some information is provided and other, matching information is to be returned by the program in response.

R

read, read access
: Interacting with a name by obtaining its associated value, or with an object by reading the value(s) of one or more of its fields, or with an input stream or other resource by obtaining the next value from it.

real time
: When things happen. On a human time scale (or faster). Contrast batch.

recipe
: The instructions for how to do something. A class is a recipe for the behavior of its instances. A constructor is the recipe for how to make an instance of its class.

recovery
: Also recovering from error. What a program ought to do after something has gone wrong; patch things up as well as possible and move on. If things are disastrous enough (e.g., after a catastrophic failure), this can be a significant task. It is facilitated by design that anticipates the need for eventual recovery.

recursion
: The use of recursive definitions to accomplish real things. Contrast just going around in circles.

recursive call
: The recursive case of a recursive function.

recursive case
: In a recursive definition, the part that relies on (a simpler form of) the thing being defined. Contrast base case.

recursive definition
: A definition in terms of itself. That is, the definition uses the thing being defined. Consists of one or more base cases and at least one recursive case. See recursive function, recursive structure.

recursive function
: A method that is defined recursively. That is, the implementation of the method contains an invocation of the same method with simpler argument(s). See recursive definition.

recursive structure
: In Java, a class whose instances contain members of the same type as the instances themselves. That is, the class defines one or more non-static fields whose type is the same as the type of the class. See recursive definition.

reference type
: The formal term for the types named by a label name.

requirements
: Expectations that a program must meet. Often identified by a designer using a technique such as use cases and embodied in a program's promises or guarantees.

reserved word
: A word that cannot be used as an identifier in Java, typically because it is a keyword.

resource library
: A class that exists to hold methods that don't logically belong to any particular object, or other (typically system-wide) resources. Typically not an instantiable class. See the chapter on Designing with Objects.

return
: A statement whose execution causes normal termination of the execution of a method body. If the return statement contains an expression, its type must match the return type of the method. In this case, the expression is evaluated prior to exiting the method body and the value of this expression is the return value of the method invocation.

return path
: See call path.

return type
: The type of the value returned by a method invocation. The first item in a method declaration.

return value
: The value returned by a method invocation.

rule
: A proto-method. Consists of a specification and a body. See the chapter on Statements and Rules.

rule body
: The set of statements detailing how a rule is to be accomplished. A proto-method body. See the chapter on Statements and Rules.

rule specification
: The information needed and provided by a rule. A proto-signature. See the chapter on Statements and Rules.

run time
: The time at which a program is executed. Run time information refers to information that is not known until the program is executed, i.e., cannot be determined from the source code alone. Contrast compile time.

runnable
: Executable, especially (`Runnable`) by a Thread.

S

scope
: The expanse of code within which a name has meaning, i.e., is a valid expression. See the note on Scoping in the chapter on Expressions. Not quite.

scribe
: One who writes.

selector
: The formal name for a getter method.

self-animating object
: An object or entity with its own animacy, i.e., one that runs concurrently and persistently. See the chapter on Self-Animating Objects.

semantics
: The rules defining what expressions and statements in a language mean (or what they do). Contrast syntax.

semicolon
: ; Used to end a simple statement.

sequence
: Two or more statements to be executed one after the other, in order. Contrast conditional, loop.

sequential
: One after another; one at a time; in order. Contrast concurrent.

server
: With respect to some service, the (computational) entity that provides that service. Contrast client.

server push
: A communication pattern in which the server initiates the service. Contrast client pull.

setter, setter method
: A method that exists solely to provide write access to a field, i.e., to change its value. Formally called a mutator.

shared reference
: A situation in which two label names refer to the same object.

shoebox name
: Also dial name. A name capable of referring to something of a primitive type, whose value is encoded directly in the memory reserved by the name. The types named by shoebox names are formally called value types. See the chapter on Things, Types, and Names.

side effect
: A change to something that occurs as a consequence of evaluating an expression. For example, an assignment.

signature
: See method signature.

slash
: / Used to delineate comments and as the division operator.

software
: Another term for computer program.

software engineer
: A person who designs, builds, tests, debugs, and maintains computer programs.

software lifecycle
: The stages of a computer program's life: design, build, test, and then maintain/adapt/modify for a very long time. Note that the vast majority of the software lifecycle is spent in the final phase(s). @@

soundness
: A promise made by a system that all information that it supplies is true (or relevant). Trivially (and not very interestingly) accomplished by providing no information at all. Contrast completeness.

source code
: See code.

standard input
: The stream which reads from the Java console. Bound to `java.lang.System.in`.

standard output
: The stream which writes to the Java console. Bound to `java.lang.System.out`.

state
: What is true of a program or entity at a specific time. Especially the current set of associations of values with names.

statement
: A piece of executable Java code. Has neither type nor value. Contrast expression. See the chapter on Statements and Rules.

static
: A modifier indicating a member of a class (rather than of its instances).

stream
: A persistent Java object which permits the reading or writing of multiple sequential values. Represents a connection to another (potentially non-Java) entity. Used for input or output.

string
: A sequence of characters. In Java, represented by the object type `String`. Although there is no primitive type representation of strings in Java, they are described in the sidebar on Java Primitive Types in the chapter on Things, Types, and Names.
Also, what's strung taut between two tin cans, carrying the sound by vibrating, in a tin can telephone.

structural recursion
: A form of recursion in which a class is defined recursively. See recursive structure. Contrast functional recursion.

style
: What we all wish we had.

subclass
: A class that inherits from another, i.e., `extends` that other. Contrast superclass. See the chapter on Inheritance.

superclass
: A class that is inherited from by another, i.e., the other `extends` the superclass. Contrast subclass. See the chapter on Inheritance.

symbolic constant
: A name, associated with an unchanging but meaningless value. Used when the uniqueness and consistency of the value are important, but the particular value is not. See the chapter on Dispatch.

syntax
: The rules defining what is legal in a certain language. Where to put the semicolons. Contrast semantics.

T

target
: In a method invocation expression, the object whose method it is.

termination condition
: The formal name for an exit condition.

test
: 1. A crucial part of program development in which program behavior is exercised in an attempt to find failures, or bugs.

: 2. In a conditional statement, another name for the boolean expression known as the condition.

Thing
: The nouns of Java, including Things of primitive type and objects. See the chapter on Things, Types, and Names.

`this`
: A Java (label) name that is bound to the current instance. Because it refers to an instance, static members are outside of its scope .

`Thread`
: A Java instruction follower.

`throw` statement
: A particular kind of Java statement, typically used with exceptions, that causes an object to be thrown and thereby circumvents the typical return trajectory. See the chapter on Exceptions.

`throws` clause
: The part of a method signature which specifies any exceptions thrown by that method. See the chapter on Exceptions.

tin can
: What one person talks into, and another listens on, in a tin can telephone. Corresponds to a socket.

  Also a container for food, though neither peanut butter nor jelly. Said food must be eliminated before construction of the telephone.

tin can telephone
: A device consisting of two tin cans, empty of food and with one end of each removed completely, and a string strung taut between a hole punched in the intact ends of each of the cans. Communication is accomplished when one person speaks into one tin can and another person listens at the other.

top down, top-down design
: An approach to design that starts by identifying the highest level, most abstract, or largest things in your system and proceeds by decomposing them. Contrast bottom up.

top level
: Immediately inside the containing structure. Top level within a class means inside the class body but not inside any other structure.

trinary operator
: An operator that takes three operands. (Also ternary operator.) See the chapter on Expressions.

type
: A partial specification of the Thing. In Java, a type is either a primitive type or an object type. See the chapter on Things, Types, and Names.

type-of-thing name-of-thing rule
: The rule that says: to declare a name, first state its type, then state its name.

typewriter
: An archaic device vaguely resembling a keyboard attached directly to a printer with no intervening memory. Requires paper.

U

unary operator
: An operator that takes one operand. See the chapter on Expressions.

unbound
: The state of a label name when it is not associated with an object, i.e., has no object referent. In this case, the label name is associated with the non-value `null`.

Unicode
: The representation used by Java for characters.

up cast
: A cast or coercion from subclass to superclass. Always valid.

use
: n. The incorporation of a resource into a program. Contrast implementation.

use case
: A description of a single interaction between a user and an entity. A technique used by a designer to identify requirements on the entity. Includes preconditions and postconditions. @@

user
: 1. A human being, with respect to a computer program.

: 2. A piece of code, with respect to another piece of code, especially an interface. Contrast implementor.

user interface
: The portion of a program with which a (human) user interacts. See also graphical user interface.

V

value
: Either a primitive value or an object.

value type
: The formal term for the types named by a shoebox name.

variable
: A Java name that has scope only from its declaration to the end of the enclosing block. Variables are formally called local variables; sometimes, this is abbreviated to locals.

vertical bar
: | Also called pipe. Used in disjunction.

view
: (In a GUI.) An object implementing how the mechanism looks. In java.awt, this typically includes its basic on-screen behavior. Contrast model.

virtual field
: A piece of state within an object that is not stored directly as a field, but is instead calculated using the values of other fields of the object. Must be accessed using a getter method as there is no field to read directly.

virtual machine
: The utility that actually runs your (compiled) Java program.

visibility
: Whether a class, field, method,field, or constructor can be used by a particular piece of code. Visibility levels include private, protected, default (or package), and public.

`void`
: The return type of a method whose invocation does not return anything. Contrast `null`.

W

web server
: A computer that provides (access to) web pages for other computers on a network.

white bread
: A substance resembling styrofoam, but with less taste and texture. Generally available in uniform white squares with pale brown edges, called crusts, that must be removed before serving to small children. Useful mostly to keep the peanut butter and jelly from getting on your fingers.

white space
: Tabs, spaces, carriage returns, and other characters that are meant to be seen as empty space.

widening, widening coercion
: Treating a thing of one type as though it were of another, larger, more precise type. Includes coercion to a larger primitive type (e.g., short to int) as well as coercion to a subclass (or super-interface) type.

write, write access
: Interacting with a name by changing its associated value, or with an object by changing the value of one or more of its fields, or with an output stream or other resource by providing a value to it.

X

Y

Z

Questions or comments:
<webmaster@cs101.org>