The Glossary

A

absolute filename
A filename whose full path is unambiguously given starting from the top (root) of a file system tree. For instance c:\Java\bin\javac.exe
See relative filename.

abstract class
A class with the abstract reserved word in its header. Abstract classes are distinguished by the fact that you may not directly construct objects from them using the new operator. An abstract class may have zero or more abstract methods.

abstraction
A simplified representation of something that is potentially quite complex. It is often not necessary to know the exact details of how something works, is represented or is implemented, because we can still make use of it in its simplified form. Object-oriented design often involves finding the right level of abstraction at which to work when modeling real-life objects. If the level is too high, then not enough detail will be captured. If the level is too low, then a program could be more complex and difficult to create and understand than it needs to be.

abstract method
A method with the abstract reserved word in its header. An abstract method has no method body. Methods defined in an interface are always abstract. The body of an abstract method must be defined in a sub class of an abstract class, or the body of a class implementing an interface.

Abstract Windowing Toolkit
The Abstract Windowing Toolkit (AWT) provides a collection of classes that simplify the creation of applications with graphical user interfaces. These are to be found in the java.awt packages. Included are classes for windows, frames, buttons, menus, text areas, and so on. Related to the AWT classes are those for the Swing packages.

accessor method
A method specifically designed to provide access to a private attribute of a class. By convention, we name accessors with a get prefix followed by the name of the attribute being accessed. For instance, the accessor for an attribute named speed would be getSpeed. By making an attribute private, we prevent objects of other classes from altering its value other than through a mutator
Accessors are used both to grant safe access to the value of a private attribute and to protect attributes from inspection by objects of other classes. The latter goal is achieved by choosing an appropriate visibility for the accessor.

actor
See client.

actual argument
The value of an argument passed to a method from outside the method. When a method is called, the actual argument values are copied into the corresponding formal arguments. The types of the actual arguments must be compatible with those of the formal arguments.

actual parameter
See actual argument.

address space
The area of virtual memory in which a process is run.

agent
See server.

aggregation
A relationship in which an object contains one or more other subordinate objects as part of its state. The subordinate objects typically have no independent existence separate from their containing object. When the containing object has no further useful existence, neither do the subordinate objects. For instance, a gas station object might contain several pump objects. These pumps will only exist as long as the station does. Aggregation is also referred to as the has-a relationship, to distinguish it from the is-a relationship, which refers to inheritance.

aliases
Multiple references to a single object. Messages may be sent to the object via any of its aliases. A resulting state change will be detectable by all.

anonymous array
An array created without an identifier. An anonymous array is usually created as an actual argument, for instance

```java
// Create an anonymous array of integers.
YearlyRainfall y2k = new YearlyRainfall(
    new int[]{ 10,10,8,8,6,4,4,0,4,4,7,10,});
```

An anonymous array may also be returned as a method result.

anonymous class
A class created without a class name. Such a class will be an sub class or an implementation of an interface, and is usually created as an actual argument or returned as a method result. For instance

```java
quitButton.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        System.exit(0);
    }
});
```

anonymous object
An object created without an identifier. They are usually created as array elements, actual arguments or method results. For instance

```java
private Point[] vertices = {
    new Point(0,0),
    new Point(0,1),
    new Point(1,1),
```
new Point(1,0),
);

See anonymous class, as these often result in the creation of anonymous objects.

API

See application programming interface.

append mode

A file writing mode, in which the existing contents of a file are retained when the file is opened. New contents are appended to the existing.

applet

Applets are Java programs based around the Applet or JApplet classes. They are most closely associated with the ability to provide active content within Web pages. They have several features which distinguish them from ordinary Java graphical applications, such as their lack of a user-defined main method, and the security restrictions that limit their abilities to perform some normal tasks.

application

Often used simply as a synonym for program. However, in Java, the term is particularly used of programs with a Graphical User Interface (GUI) that are not applets.

application programming interface (API)

A set of definitions that you can make use of in writing programs. In the context of Java, these are the packages, classes, and interfaces that can be used to build complex applications without having to create everything from scratch.

argument

Information passed to a method. Arguments are also sometimes called parameters. A method expecting to receive arguments must contain a formal argument declaration for each as part of its method header. When a method is called, the actual argument values are copied into the corresponding formal arguments.

arithmetic expression

An expression involving numerical values of integer or floating point types. For instance, operators such as +, -, *, / and % take arithmetic expressions as their operands and produce arithmetic values as their results.

arithmetic operator

Operators, such as +, -, *, / and %, that produce a numerical result, as part of an arithmetic expression.

Arpanet

A network that was a forerunner of the global Internet.

array

A fixed-size object that can hold zero or more items of the array's declared type.

array initializer

An initializer for an array. The initializer takes the place of separate creation and initialization steps. For instance, the initializer

```
int[] pair = { 4, 2, };
```

is equivalent to the following four statements.

```
int[] pair;
pair = new int[2];
pair[0] = 4;
```
pair[1] = 2;

assembler
The program used to translate a program written in assembly language into the binary form of a particular instruction set.

assembly language
A symbolic language corresponding closely to the instruction set of a Central Processing Unit. The program used to translate a program written in assembly language is called an assembler.

assignment operator
The operator (=) used to store the value of an expression into a variable, for instance:

variable = expression;

The right-hand-side is completely evaluated before the assignment is made. An assignment may, itself, be used as part of an expression. The following assignment statement stores zero into both variables.

x = y = 0;

assignment statement
A statement using the assignment operator.

attribute
A particular usage of an instance variable. The set of attribute values held in a particular instance of a class define the current state of that instance. A class definition may impose particular constraints on the valid states of its instances by requiring that a particular attribute, or set of attributes, do not take on particular values. For instance, attributes holding coursework marks for a class should not hold negative values. Attributes should be manipulated by accessor and mutator methods.

B

base case
A non-recursive route through a recursive method.

base type
The type of items which may be stored in an array - the array's defined type. For instance, in

int[] numbers;
the base type of numbers is int. Where the base type is a class type, it indicates the lowest super type of objects that may be stored in the array. For instance, in

Ship[] berths;
only instances of the Ship class may be stored in berths. If the base type of an array is Object, instances of any class may be stored in it.

behavior
The methods of a class implement its behavior. A particular object's behavior is a combination of the method definitions of its class and the current state of the object.

big-endian
A common difference between machines is the order in which they store the individual bytes of multi-byte numerical data. A big-endian machine stores the higher-order bytes before the lower-order bytes. See little-endian.

binary
Number representation in base 2. In base 2, only the digits 0 and 1 are used. Digit positions represent successive powers of 2. See bit.

binary operator
An operator taking two operands. Java has many binary operators, such as the arithmetic operators +, -, *, / and %, and the boolean operators &&, || and ^, amongst others.

binary search
A search of sorted data, in which the middle position is examined first. Search continues with either the left or the right portion of the data, thus eliminating half the remaining search space. This process is repeated at each step, until either the required item is found, or there is no more data to search.

bit
A binary digit, which can take on two possible values: 0 and 1. Bits are the fundamental building block of both programs and data. Computers regularly move data around in multiples of eight-bit units (bytes for the sake of efficiency).

bit manipulation operator
Operators, such as &, | and ^, that are used to examine an manipulate individual bits within the bytes of a data item. The shift operators, <<, >> and >>>, are also bit manipulation operators.

blank final variable
A final variable that is not initialized as part of its declaration. Such a variable must be initialized in either an instance initialization block or all of the constructors for its class before it is used. A static blank final variable must be initialized in a static initialization block.

block
Statements and declarations enclosed between a matching pair of curly brackets ({}). For instance, a class body is a block, as is a method body. A block encloses a nested scope level.

bookmark
Used by a Web browser to remember details of a Uniform Resource Locator (URL).

boolean
One of Java's primitive types. The boolean type has only two values: true and false.

boolean expression
An expression whose result is of type boolean, i.e. gives a value of either true or false. Operators such as && and || take boolean operands and produce a boolean result. The relational operators take operands different types and produce boolean results.

boot
When a computer is switched on it is said to 'boot up'. This term comes from the phrase, 'Pulling yourself up by your bootstraps.' Before a computer is ready to be used, it must load the programs that it needs from its disks, but this means that it must have a program of some sort available in order to be able to load the programs it needs! The loading program is called a bootstrap.

bootstrap classes
Classes that make up the Java Platform Core Application Programming Interface (API), such as those found in the java.lang, java.io and java.io packages.
boundary error
Errors that arise from programming mistakes made at the edges of a problem - indexing off the edge of an array, dealing with no items of data, loop termination and so on. Boundary errors are a very common type of logical error.

bounded repetition
Repetition where the statements within a loop's body are performed a fixed number of times and the number of times is established when the loop is started. There is no control structure in Java that guarantees bounded repetition. See unbounded repetition.

bounds
The limits of an array or collection. In Java, the lower limit is always zero. In the case of an array, the upper bound is one less than then length of the array, and is fixed. Indexing outside the bounds of an array or collection will result in an IndexOutOfBoundsException exception being thrown.

branch instruction
Stores a new instruction address into the program counter. The effect of this is the next instruction to be fetched will not usually be the one immediately following the branch instruction. Hence the normal sequential execution of instructions is disrupted. This allows both repetition and conditional execution of instructions to be effected.

break statement
A statement used to break out of a loop, switch statement or labeled block. In all cases, control continues with the statement immediately following the containing block.

bridging method
A method that provides a bridge between the methods of a class's public interface and its private implementation. Bridging methods will typically have non-public visibility.

byte
In general computing, this refers to eight bits of data. In Java it is also the name of one of the primitive data types, who size is eight bits.

bytecode
Java source files are translated by a compiler into bytecodes - the instruction set of the Java Virtual Machine (JVM). Bytecodes are stored in .class files.

C

call-by-value
A semantics of passing an argument to a method in which a copy of the actual argument value is taken and placed in a separate memory location, represented by the corresponding formal argument. As a result, assignment to a formal argument within a method can have no effect on the value stored in the actual argument. This principle is often misunderstood in Java. It does not mean that an object referred to by an actual argument cannot be modified via the formal argument. Consider the following example of sorting the array referred to by the variable numbers
Arrays.sort(numbers);
The sort method will change the order of the values stored in the object referred to by numbers. However, it is impossible for the sort method to
change which array numbers refers to - a sorted copy, for instance. Some languages provide an argument passing semantics known as call-by-reference, in which an actual argument's value may be changed. Java does not provide this, however.

carriage return
The \r character. Also used as a synonym for the 'Return' or 'Enter' key used to terminate a line of text. The name derives from the carriage on a mechanical typewriter.
cascading if-else statement
A form of if-else statement in which each else-part (except the last) consists of a further nested if-else statement. Used to overcome the problem of textual drift often associated with nested if statements.
case label
The value used to select a particular case in a switch statement.
case sensitive
A test that is sensitive to whether a character is upper-case (e.g., 'A') or lower-case (e.g., 'a').
cast
Where Java does not permit the use of a source value of one type, it is necessary to use a cast to force the compiler to accept the use for the target type. Care should be taken with casting values of primitive types, because this often involves loss of information. Casts on object references are checked at runtime for legality. A ClassCastException exception will be thrown for illegal ones.
catch clause
The part of a try statement responsible for handling a caught exception.
catching exceptions
Exceptions are caught within the catch clause of a try statement. Catching an exception gives the program an opportunity to recover from the problem or attempt a repair for whatever caused it.
Central Processing Unit
The Central Processing Unit (CPU) is the heart of a computer as it is the part that contains the computer's ability to obey instructions. Each type of CPU has its own instruction set.
character set encoding
The set of values assigned to characters in a character set. Related characters are often grouped with consecutive values, such as the alphabetic characters and digits.
checked exception
An exception that must be caught locally in a try statement, or propagated via a throws clause defined in the method header. See unchecked exception.
class
A programming language concept that allows data and methods to be grouped together. The class concept is fundamental to the notion of an object-oriented programming language. The methods of a class define the set of permitted operations on the class's data (its attributes). This close tie between data and operations means that an instance of a class - an object - is responsible for responding to messages received via its defining class's methods.
class body

Object Oriented-Programming with Java: An Introduction
The body of a class definition. The body groups the definitions of a class's members - fields, methods and nested classes.

class constant
A variable defined as both final and static.

class header
The header of a class definition. The header gives a name to the class and defines its access. It also describes whether the class extends a super class or implements any interfaces.

class inheritance
When a super class is extended by a sub class, a class inheritance relationship exists between them. The sub class inherits the methods and attributes of its super class. In Java, class inheritance is single inheritance. See interface inheritance for an alternative form of inheritance.

class method
A synonym for static method.

classpath
The path searched by the compiler and interpreter for class definitions. The class path may be set by a command-line argument to either, or via an environment variable.

class scope
Private variables defined outside the methods within a class have class scope. They are accessible from all methods within the class, regardless of the order in which they are defined. Private methods also have class scope. Variables and methods may have a wider scope if they do not use the private access modifier.

class variable
A synonym for static variable.

client
The user of a service. A Web client requests resources from a Web server, for instance. When the client is an object, it is the sender of messages to its object servers.

cohesion
The extent to which a component performs a single well-defined task. A strongly cohesive method, for instance, will perform a single task, such as adding an item to a data structure, or sorting some data, whereas a weakly cohesive method will be responsible for several disparate tasks. Weakly cohesive components should, in general, be split into separate more cohesive components. The java.util package is a weakly cohesive package because it contains many unrelated classes and interfaces, whereas the java.io package is highly cohesive.

command-line argument
Arguments passed to a program when it is run. A Java program receives these in the single formal argument to its main method

```java
public static void main(String[] args)
```

The arguments are stored as individual strings.

comment
A piece of text intended for the human reader of a program. Compilers ignore their contents.

Common Gateway Interface
The Common Gateway Interface (CGI) is a standard that allows Web clients to interact with programs on a Web server. A CGI script on the server is able to process input or arguments from a client, and respond accordingly.

**compilation**
The process of translating a programming language. This often involves translating a *high level programming language* into a *low level programming language*, or the binary form of a particular *instruction set*. The translation is performed by a program called a *compiler*. A Java compiler translates programs into *bytecodes*.

**compiler**
A program which performs a process of *compilation* on a program written in a *high level programming language*.

**complement operator**
The complement operator, ~, is used to invert the value of each *bit* in a *binary* pattern. For instance, the complement of 1010010 is 0101101.

**concurrency**
A feature of *parallel programming*. Parts of a program whose executions overlap in time are said to execute concurrently. Java's *thread* feature support concurrency.

**condition**
A *boolean expression* controlling a conditional statement or loop.

**conditional operator**
An *operator* taking three operands - a ternary operator. The conditional operator (? : ) is used in the form

```
 bexpr ? expr1 : expr2
```

where *bexpr* is a *boolean expression*. The the boolean expression has the value *true* then the result of the operation is the value of *expr1*, otherwise it is the value of *expr2*.

**connection handshake**
An exchange of messages between two processes in an attempt to establish a connection between them.

**constant**
A *variable* whose value may not be changed. In Java, these are implemented by *final variables*.

**constructor**
A constructor is automatically called when an *instance* of its class is created. A constructor always has the same name as its class, and has no return type. For instance

```java
public class Ship {
    public Ship(String name){
        ...
    }
    ...
}
```

A class with no explicit constructor has an implicit *no-arg constructor*, which takes no arguments and has an empty body.

**continue statement**
A statement that may only be used inside the body of a loop. In the case of a while loop or do loop, control passes immediately to the loop's terminating test. In the case of a for loop, control passes to the post-body update expression.

**continuous simulation**
In a continuous simulation, time ticks past at a regular rate that is applicable to the particular simulation scenario. At each tick, all the objects in the simulation are informed of the passage of time and updated accordingly. See discrete simulation for an alternative form of simulation.

coupling
Coupling arises when classes are aware of each other because their instances must interact. Linkage between two classes that may be either strong or weak. Stronger coupling arises when one class has a detailed knowledge of the internal implementation of another, and is written to take advantage of that knowledge. So anything that has the potential to reduce the amount of inside knowledge will tend to weaken coupling. Hence, information hiding and encapsulation. Java's visibility levels - private, package, protected, public - progressively reveal detail to other classes, and so increase the potential for stronger coupling. Interfaces are one way to reduce to reduce coupling - because you interact with a class via an abstract definition, rather than a concrete implementation.

critical section
A section of code in which there is potential for a race hazard. Critical sections make use of the synchronized methods or statements.

decrement operator
An operator (--) that adds one to its operand. It has two forms: pre-decrement (--x) and post-decrement (x--). In its pre-decrement form, the result of the expression is the value of its argument after the decrement. In its post-decrement form, the result is the value of its argument before the decrement is performed. After the following,
```java
int a = 5, b = 5;
int y, z;
y = --a;
```
z = b--
y has the value 4 and z has the value 5. Both a and b have the value 4.
daemon thread
Daemon threads are non-user threads. They are typically used to carry out
low-priority tasks that should not take priority over the main task of the
program. They can be used to do useful work when all other user threads are
blocked. The garbage collector is one example of a daemon thread.
datagram
A packet of information passed between two communicating processes across
a network. Both the Transmission Control Protocol (TCP) and the User
Datagram Protocol (UDP) are indirectly involved in sending datagrams to
provide reliable or unreliable communication, respectively.
data type
There are eight primitive data types in Java; five of these represent numerical
types of varying range and precision - double, float, int, long and short.
The remaining three are used to representing single-bit values (boolean),
single byte values (byte) and two-byte characters from the ISO Unicode
character set (char).
deadlock
A situation that arises when two threads each acquires the lock to one of a set
of resources that they both need.
decimal
Number representation in base 10. In base 10, the digits 0 to 9 are used. Digit
positions represent successive powers of 10.
declaration and initialization
A statement in which a variable is declared and immediately given its initial
value. Three examples of declaration and initialization are
```java
int numStudents = 23;
Ship argo = new Ship();
Student[] students = new Student[numStudents];
```
Instance variables that are not explicitly initialized when they are declared
have a default initial value that is appropriate to their type. Uninitialized local
variables have an undefined initial value.
deep copy
A copy of an object in which copies of all the object’s sub-components are
also made. The resulting object might, in effect, be a clone of the original. See
shallow copy for an alternative.
default initial value
The default value of any variable not explicitly initialized when it is declared.
Fields of numeric primitive types have the value zero by default, boolean
variables have the value false, char variables have the value \u0000 and
object references have the value null. The initial values of local variables are
undefined, unless explicitly initialized.
default label
The destination for all values used in a switch statement expression that do not
have explicit case labels. A default label is optional.
delegation
The process by which an object passes on a message it has received to a sub-
ordinate object. If inheritance is not available in a programming language,
delegation is the most viable alternative for avoiding code duplication and
promoting code reuse.
De Morgan's Theorem
Two rules that can help to simplify boolean expressions involving multiple logical-not operators in combination with other boolean operators.

deprecated
Something that has been made obsolete by later versions of the API. Deprecated methods should not be used because there is no guarantee that they will continue to exist in future versions.

direct recursion
Recursion that results from a method calling itself.

discrete simulation
In a discrete simulation, time passes at an irregular rate that is determined by the primary events of interest in the simulation. See continuous simulation for an alternative form of simulation.

disk drive
A hardware device used to store data. They come in many forms, such as compact disks, floppy disks and hard disks.

divide and conquer
An approach to problem solving that attempts to reduce an overall single large problem into multiple simpler problems.

do loop
One of Java's three control structures used for looping. The other two are the while loop and for loop. A do loop consists of a loop body and a boolean expression. The condition is tested after the loop body has been completed for the first time and re-tested each time the end of the body is completed. The loop terminates when the condition gives the value false. The statements in the loop body will always be executed at least once.

dotted decimal notation
The notation used to represent the four byte values of an IP address. Each byte is represented as a value between 0 and 255, for instance 129.12.0.1. The most-significant byte is written first.

double buffering
A graphics technique used to smooth animation. The next version of an image is drawn 'behind the scenes' and then displayed in its entirety when the drawing is complete. The assumption is that it will be relatively quick to display the fully drawn image, compared to the time it takes to compute and draw it.

downcast
A cast towards an object's dynamic type - that is, 'down' the inheritance hierarchy. For instance

```java
// Downcast from Object to String
String s = (String) o;
```

See upcast.

dynamic type
The dynamic type of an object is the name of the class used to construct it. See static type.
A common part of the program development process. A source file is created initially and compiled. Syntax errors must be corrected in the editor before compiling it again. Once the program has been successfully compiled, it can be run. The program's execution might reveal logical errors, or the need for enhancements. A further edit-compile-run iteration is the result.

**encapsulation**

Safeguarding the state of an objects by defining its attributes as `private` and channeling access to them through `accessor` and `mutator` methods.

**enumerated type**

A data type - not directly available in Java - in which symbolic names are used for a sequence of constant numeric values. They facilitate the avoidance of `magic numbers`. They can be simulated in Java with fields in an interface, for instance

```java
public interface States {
    public static final int Stop = 0, Go = 1;
}
```

However, the compiler type checking usually available with enumerated types is not available with this form.

**exception**

An object representing the occurrence of an exceptional circumstance - typically, something that has gone wrong in the smooth running of a program. Exception objects are created from `classes` that extend the `Throwable` class. See **checked exception** and **unchecked exception**.

**exception handler**

The `try statement` acts as an exception handler - a place where exception objects are caught and dealt with.

**exclusive-or operator**

The exclusive-or operator (`^`) is both a `boolean operator` and a `bit manipulation operator`. The boolean version gives the value `true` if only one of its operands is `true`, otherwise it gives the value `false`. Similarly, the bit manipulation version produces a `1` bit wherever the corresponding bits in its operands are different.

**expression**

A combination of `operands` and `operators` that produces a resulting value. Expressions have a resulting type, that affects the context in which they may be used. See **boolean expression** and **arithmetic expression**, for instance.

**F**

**factory pattern**

A `pattern` of class definition that is used as a generator of `instances` of other classes. Often used to create platform- or locale-specific implementations of `abstract classes` or `interfaces`. This reduces `coupling` between classes as it frees the factory's client from a need to know about particular implementations.

**fetch-execute cycle**

The simple set of steps that are endlessly repeated by a computer's `Central Processing Unit` for each program instruction: `Fetch the next instruction referenced by the program counter,' `Update the program counter to refer to the next instruction,' `Execute the instruction just fetched.'
field

Variables defined inside a class or interface, outside of the methods. Fields are members of a class.

file system

An operating system makes it possible to use space on a computer's disk drives by imposing a structured file system on the disk storage. Each file system has its own conventions for the way in which files are named, folders and directories are structured, and large files are split into smaller pieces, for instance. It is not usually possible to transfer data directly from the file system of one operating system to that of a different operating system, because their conventions are likely to be incompatible.

File Transfer Protocol

The File Transfer Protocol (FTP) defines a standard set of rules that make it possible to transfer a file from one file system to another.

filter stream

An input-output class that filters or manipulates its stream of input- or output-data in some way. Two examples are DataInputStream and DataOutputStream.

final class

A class with the final reserved word in its header. A final class may not be extended by another class.

finalization

Immediately before an object is garbage collected, its finalize method is called. This gives it the opportunity to free any resources it might be holding on to.

finally clause

Part of a try statement that is always executed, either following the handling a caught exception, or normal termination of the protected statements.

final method

A method with the final reserved word in its header. A final method may not be overridden by a method defined in a sub class.

final variable

A variable with the final reserved word in its declaration. A final may not assigned to once it has been initialized. Initialization often takes place as part of its declaration. However, the initialization of an uninitialized final field (known as a blank final variable) may be deferred to the class's constructor, or an initializer.

first in, first out

The (FIFO) semantics of a queue data structure. Items are removed in the order in which they arrived in the queue, so older items are always removed before newer ones. See last in, first out.

floating point number

See real number.

for loop

One of Java's three control structures used for looping. The other two are the while loop and do loop. A for loop consists of a loop header and a loop body. The header consists of three expressions separated by two semicolons and one or more of these may be omitted. The first expression is only evaluated once, at the point the loop is entered. The middle expression is a boolean expression representing the loop's termination test. An empty expression represents the
value `true`. The third expression is evaluated after each completion of the loop's body. The loop terminates when the termination test gives the value `false`. The statements in the loop body might be executed zero or more times.

formal argument
The definition of a method's argument which are part of a *method header*. Each formal argument has an associated type. When a method is called, the *actual argument* values are copied into the corresponding formal arguments. The types of the actual arguments must be compatible with those of the formal arguments.

formal parameter
See *formal argument*.

fully qualified class name
The name of a class, including any *package* name and enclosing class name.

Given the following class outline

```java
package oddments;

class Outer {
    public class Inner {
        ...
    }
    ...
}
```

The fully qualified name of `Inner` is `oddments.Outer.Inner`

fully evaluating operator
An operator that evaluates all of its arguments to produce a result. Standard *arithmetic operators*, such as `+`, are fully evaluating. In contrast, some *boolean operators*, such as `&&`, are *short-circuit operators*.

functional programming
A style of programming associated with languages such as Haskell. Functional programming languages are more closely tied to a mathematical concept of 'function' than *imperative programming languages*. This makes it easier to apply program-proving techniques and logical reasoning to functional programs. In particular, functional programs do not use the concept of *variables* in the traditional sense, i.e. a memory location whose contents might be changed from time to time as a program executes.

G

garbage collector
A *daemon thread* that recycles objects to which there are no extant references within a program.

global variable
A phenomenon that is more usually regarded as being a problem in *structured programming languages* than in *object-oriented languages*. In a structured programming language, such as *Pascal* or *C*, a global variable is one defined outside the procedures and functions of a program. It is difficult to keep track of the usage of such a variable as it is readable and writable by the whole program or module in which it is defined. This makes such variables a common source of logical errors. In fact, *instance variables* pose a similar problem within class definitions, since Java's *scope rules* make them
accessible to all methods defined within a class. This is one of the reasons why we prefer to channel access to instance variables through accessor and mutator methods even within a class.

Graphical User Interface

A Graphical User Interface (GUI) is part of a program that allows user interaction via graphical components, such as menus, buttons, text areas, etc. Interaction often involves use of a mouse.

H

hardware

The physical devices of a computer system, such as its micro-chips, disk drives, keyboard, printer, sound card, and so on. It is called 'hardware' in contrast to programs, which are called 'software'.

has-a relationship

See aggregation.

hash code

A value returned by a hash function. A hash code can be used as an index into a random-access data structure, providing an efficient mapping between an object and its location. Used by classes such as HashMap.

hash function

A function used to produce a hash code from the arbitrary contents of an object. Classes can override the hashValue method, inherited from the Object class, to define their own hash function.

heterogeneous collection

A collection of objects with different dynamic types. See homogeneous collection.

hexadecimal

Number representation in base 16. In base 16, the digits 0 to 9 and the letters A to F are used. A represents 10 (base 10), B represents 11 (base 10), and so on. Digit positions represent successive powers of 16.

high level programming language

Languages such as Java, C++, Ada, etc. which provide programmers with features such as control structures, methods, classes, packages, etc. These features are largely independent of any particular instruction set, and hence programs written in these languages tend to be more portable than those written in low level programming languages.

homogeneous collection

A collection of objects with the same dynamic type. Arrays are the most common homogeneous collection objects. See heterogeneous collection.

hostname

The name of a host system.

host system

A computer system on which a process is run.

hot spot

An area in an image map with a particular significance. A program typically monitors movements of the mouse, and responds according to the actions associated with the hot spots over which it passes. This might include displaying different status information, for instance. Often, clicking the mouse on a hot spot is used to indicate that the program should activate an associated

---

Object Oriented-Programming with Java: An Introduction
action. The term *hot spot* is also used to signify a computationally intensive part of a program, such as an inner loop. Such places are often a potential target for program optimization.

**HSB Color Model**
A color model based upon representing a color as three components: hue, saturation and brightness. This is sometimes known as the HSV color model - hue, saturation and value. See *RGB Color Model*.

**HyperText Markup Language**
The HyperText Markup Language (HTML) is a simple presentation language used to markup the content of Web pages. Its *tags* often appear in pairs to mark sections of text that should be represented in different colors of fonts.

**HyperText Transfer Protocol**
The HyperText Transfer Protocol (HTTP) is a set of rules defined to enable a Web client (browser) to interact with a Web server.

**I**

**icon**
An image intended to communicate a language- or culturally-independent meaning.

**identifier**
A programmer-defined name for a *variable, method, class or interface*.

**IEEE 754**
Standard 754-1985 issued by the Institute of Electrical and Electronic Engineers for binary floating point arithmetic. This is the standard to which Java's arithmetic conforms.

**if-else statement**
A *control structure* used to choose between performing one of two alternative actions.

```java
if(boolean-expression){
    // Statements performed if expression is true.
    ...
}
else{
    // Statements performed if expression is false.
    ...
}
```
It is controlled by a *boolean expression*. See *if statement*.

**if statement**
A *control structure* used to choose between performing or not performing further actions.

```java
if(boolean-expression){
    // Statements performed if expression is true.
    ...
}
```
It is controlled by a *boolean expression*. See *if-else statement*.

**image map**
An image divided into logical areas, each of which has a *hot spot*.

**immutable object**
An object whose *state* may not be changed. Objects of the String class are immutable, for instance - their length and contents are fixed once created.

**imperative programming**
The style of programming usually associated with languages such as C, Fortran, Pascal and so on. Imperative programming is distinguished from functional programming in that the former is strongly tied to the concept of variables and memory locations. A variable is associated with a memory location and the contents of that memory location may be changed, via the variable, over the course of time. The meaning or effect of a program fragment at a particular point can only be understood by reference to the current contents of the set of relevant variables, therefore. In contrast, functional programs do not allow the contents of a variable to be changed once set (in simplified terms), hence making them easier to reason about. While languages such as C++ and Java are also imperative programming languages, strictly speaking, they are more commonly referred to as object-oriented programming languages.

**implements clause**
That part of of a class header that indicates which interfaces are implemented by the class. A class may implement more than one interface. See multiple inheritance.

**implicit type conversion**
Type conversion that does not require a cast. Implicit type conversions typically do not involve any loss of information. For instance, combining an integer operand with a floating point operand in an arithmetic expression will result in an implicit type conversion of the integer to an equivalent floating point value.

**import statement**
A statement that makes the names of one or more classes or interfaces available in a different package from the one in which they are defined. Import statements follow any package declaration {package!declaration}, and precede any class or interface definitions.

**inconsistent state**
A state that an object should not be in. A class needs to be carefully designed in order to ensure that none of its instances can get into an inconsistent state. An example of an inconsistent state might be a football team with too many players on the field.

**increment operator**
An operator (++) that adds one to its operand. It has two forms: pre-increment (++x) and post-increment (x++). In its pre-increment form, the result of the expression is the value of its argument after the increment. In its post-increment form, the result is the value of its argument before the increment is performed. After the following,

```java
int a = 5, b = 5;
int y, z;
y = ++a;
z = b++
```

y has the value 6 and z has the value 5. Both a and b have the value 6.

**indirect recursion**
Recursion that results from method Y calling method X, when an existing call from X to Y is still in progress.

**infinite loop**
A loop whose termination test never evaluates to false. Sometimes this is a deliberate act on the part of the programmer, using a construct such as

```java
while(true) ...
```

---

Object Oriented-Programming with Java: An Introduction
or

\begin{verbatim}
for( ; ; ) ...
\end{verbatim}

but it can sometimes be the result of a logical error in the programming of a normal loop condition or the statements in the body of the loop.

infinite recursion
Recursion that does not terminate. This can result from any of direct recursion, indirect recursion or mutual recursion. It is usually the result of a logical error, and can result in stack overflow.

information hiding
The practice of ensuring that only as much information is revealed about the implementation of a class as is strictly required. Hiding unnecessary knowledge of implementation makes it less likely that other classes will rely on that knowledge for their own implementation. This tends to reduce the strength of coupling between classes. It also reduces that chance that a change of the underlying implementation will break another class. Ensuring that all fields of a class are defined as private, is one of the ways that we seek to promote information hiding.

inheritance
A feature of object-oriented programming languages in which a sub type inherits methods and variables from its super type. Inheritance is most commonly used as a synonym for class inheritance {class!inheritance}, but interface inheritance is also a feature of some languages, including Java.

inheritance hierarchy
The relationship between super classes and sub classes is known as an inheritance hierarchy. Single inheritance of classes means that each class has only a single `parent' class and that the Object class is the ultimate ancestor of all classes - at the top of the hierarchy. Two classes that have the same immediate super class can be thought of as sibling sub classes. Multiple inheritance of interfaces gives the hierarchy a more complex structure than that resulting from simple class inheritance.

initializer
A block defined at the outermost level of a class - similar to a method without a header. Initializer blocks are executed, in order, when an instance is created. They are executed before the constructor of the defining class, but after any super class constructor. They are one of the places in which blank final variables may be initialized.

inner class
A class defined inside an enclosing class or method. We use the term to refer to non-static nested classes.

instance
A synonym for object. Objects of a class are instantiated when a class constructor is invoked via the new operator.

instance variable
A non-static field of a class. Each individual object of a class has its own copy of such a field. This is in contrast to a class variable which is shared by all instances of the class. Instance variables are used to model the attributes of a class.

instantiation
The creation of an instance of a class - that is, an object.

instruction set
The set of instructions that characterize a particular *Central Processing Unit*. Programs written in the instruction set of one type of CPU cannot typically be run on any other type of CPU.

**integer**
A positive or negative whole number. The *primitive types* `byte`, `short`, `int` and `long` are used to hold integer values within narrower or wider ranges.

**interface inheritance**
When a *class* implements an *interface*, an interface *inheritance* relationship exists between them. The class inherits no implementation from the interface, only method signatures and *static variables*. It is also possible for one interface to extend one or more interfaces. In Java, interface inheritance is the only form of *multiple inheritance*. See *class inheritance* for an alternative form of inheritance.

**Internet**
A global network of many interconnected networks.

**Internet Service Provider**
An Internet Service Provider (ISP) provides connections to the *Internet* for users who do not have their own network. The ISP provides such user with their own *IP address* that enables them to interact with other computers attached to the Internet.

**interpretational inner class**
An *inner class* whose role is to provide a view or interpretation of data belong to its enclosing class, but independent of the data's actual representation.

**interpreter**
A program which executes a translated version of a source program by implementing a virtual machine. Interpreters typically simulate the actions of an idealized *Central Processing Unit*. An interpreter for Java must implement the *Java Virtual Machine (JVM)* and executes the *bytecodes* produced by a Java *compiler*. The advantage of using an interpreter for Java is that it make the language more *portable* than if it were a fully compiled language. The bytecode version of a program produced by a Java compiler may be run on any interpreter implementing the JVM.

**interprocess communication**
The ability of two or more separate *processes* to communicate with one another.

**interrupt**
An asynchronous message sent to a *process or thread* that interrupts what it is currently doing. This usually results in an *InterruptedException* object being received by the interrupted thread. Waiting for an interrupt is an alternative to *polling*.

**IP address**
An Internet Protocol (IP) address for a networked computer. Currently, IP addresses consist of four byte values, written in *dotted decimal notation*, such as 129.12.0.1. In future, IP addresses will be sixteen bytes long to accommodate the expansion in the number of networked computers.

**is-a relationship**
See *inheritance*.

**iteration**
Repetition of a set of statements, usually using a looping *control structure*, such as a *while loop*, *for loop* or *do loop*. 

*Object Oriented-Programming with Java: An Introduction*
iterator pattern
A common pattern in which the contents of a collection are iterated over in order. The Iterator pattern frees a client of the data from needing details of the how the data is stored. This pattern is supported by the Iterator and ListIterator interfaces.

J

Java
A portable high level programming language released by Sun Microsystems.
Java Archive file
A Java Archive (JAR) file makes it possible to store multiple bytecode files in a single file.
Java 2 SDK
A particular implementation of the abstract functionality described in Sun's specification of the Java 2 Platform.
Java Virtual Machine (JVM)
An idealized machine whose instruction set consists of bytecodes. A Java program is compiled to an equivalent bytecode form and executed on an interpreter which implements the JVM.

K

key value
The object used to generate an associated hash code for lookup in an associative data structure.

L

last in, first out
The (LIFO) semantics of a stack data structure. Items are removed in the opposite order to which they arrived in the stack, so newer items are always removed before older ones. See first in, first out.
layout manager
An object responsible for sharing the available space between multiple components within a graphical container.
left shift operator
The left shift operator (<<) is a bit manipulation operator. It moves the bits in its left operand zero or more places to the left, according to the value of its right operand. Zero bits are added to the right of the result.
lexicographic ordering
The ordering of words as they would be found in a dictionary. It should be noted that different locales order similar looking words according to their own conventions - this applies, in particular, to accented characters.
little-endian
A common difference between machines is the order in which they store the individual bytes of multi-byte numerical data. A little-endian machine stores the lower-order bytes before the higher-order bytes. See big-endian.

**livelock**

A situation in which a *thread* waits to be notified of a condition but, on waking, finds that another thread has inverted the condition again. The first thread is forced to wait again. When this happens indefinitely, the thread is in *livelock.*

**local inner class**

*An inner class* defined within a method.

**local variable**

A variable defined inside a *method body.*

**locale**

Details which are dependent upon conventions and customs adopted by a particular country or culture. Within programs, this affects issues such as number and date formatting, for instance. Designers of classes should be sensitive to the locale-specific issues that might apply to users.

**logical error**

An error in the logical of a method or class. Such an error might not lead to an immediate *runtime error,* but could have a significant impact on overall program correctness.

**logical operators**

Operators, such as `&&`, `||`, `&`, `|` and `^` that take two boolean operands and produce a boolean result. Used as part of a *boolean expression,* often in the condition of a *control structure.*

**look-and-feel**

The visual impression and interaction style provided by a user interface. This is predominantly the responsibility of the *window manager* (in collaboration with the underlying *operating system*) running on a particular computer. It refers to style of such things as window title bars, how windows are moved and resized, how different operations are performed via a mouse, and so on. It is preferable to have a consistent look and feel within a single user environment. However, some window managers do allow individual programs to present a different look and feel from the predominant style of the host environment. Java's Swing components support this idea by allowing an application to select a 'pluggable look and feel' from those provided by a user interface manager. An application running in a Microsoft Windows environment could be made to look like one that normally runs in an X Windows environment, for instance. This allows an application to look similar on different platforms, but it can also lead to confusion for users.

**loop variable**

*A variable* used to control the operation of a loop, such as a *for loop.* Typically, a loop variable will be given an initial value and it is then incremented after each *iteration* until it reaches or passes a terminating value.

**low level programming languages**

Often known as 'assembly languages', these provide little more than the basic instruction set of a particular *Central Processing Unit.* Hence programs written in low level programming languages tend to be less *portable* than those written in *high level languages.*
M

magic number
A constant value with a significance within a particular context. For instance, the value 12 could mean many different things - the number of hours you have worked today, the number of dollars you are owed by a friend, and so on. As far as possible, such values should be associated with an identifier that clearly expresses their meaning.

```java
final int maxSpeed = 50;
```

If stored in a final variable, it is unlikely that any execution overhead will be incurred by doing so.

main method
The starting point for program execution

```java
public static void main(String[] args)
```

manifest file
A file held in a Java Archive (JAR) file, detailing the contents of the archive.

marking interface
An interface with no methods.

member
The members of a class are fields, methods and nested classes.

memory leak
A situation in which memory that is no longer being used has not been returned to the pool of free memory. A garbage collector is designed to return unreferenced objects to the free memory pool in order to avoid memory leaks.

message passing
We characterize object interactions as message passing. A client object sends a message to a server object by invoking a method from the server's class. Arguments may be passed with the message, and a result returned by the server.

method
The part of a class definition that implements some of the behavior of objects of the class. The body of the method contains declarations of local variables and statements to implement the behavior. A method receives input via its arguments, if any, and may return a result if it has not been declared as void.

method body
The body of a method: everything inside the outermost block of a method.

method header
The header of a method, consisting of the method name, its result type, formal arguments and any exceptions thrown. Also known as a method signature.

method overloading
Two or more methods with the same name defined within a class are said to be overloaded. This applies to both constructors and other methods. Overloading applies through a class hierarchy, so a sub class might overload a method defined in one of its super classes. It is important to distinguish between an overloaded method and an overridden method. Overloaded methods must be distinguishable in some way from each other; either by having different numbers of arguments, or by the types of those arguments being different. Overridden methods have identical formal arguments.

method overriding

Object Oriented-Programming with Java: An Introduction
A method defined in a *super class* may be overridden by a method of the same name defined in a *sub class*. The two methods must have the same name and number and types of formal arguments. Any checked exception thrown by the sub class version must match the type of one thrown by the super class version, or be a sub class of such an exception. However, the sub class version does not have to throw any exceptions that are thrown by the super class version. It is important to distinguish between method overriding and *method overloading*. Overloaded methods have the same names, but differ in their formal arguments. See *overriding for breadth*, *overriding for chaining* and *overriding for restriction*.

**method result**

The value returned from a method via a *return statement*. The type of the expression in the return statement must match the return type declared in the *method header*.

**method signature**

A synonym for *method header*.

**method variable**

See *local variable*.

**micro-chip**

A small electronic device used to build computers and other electronic equipment. Chips are commonly used to supply the memory and processing components of a computer. See *Central Processing Unit*.

**MIME**

Multipurpose Internet Mail Extensions (MIME) are rules that make it possible to use electronic mail to send content other than simple text.

**modal**

A dialog is modal if its parent application is blocked from further activity until the dialog has completed. See *non-modal*.

**model-view pattern**

A *pattern* in which the representation of data (the model) is kept separate from its visualization (the view). Such decoupling makes it easier to change the underlying data representation, or provide multiple views, for instance. Quite often a third element is added to create the Model-View-Controller (MVC) pattern. In the MVC pattern, those elements of the system which are able to control or modify the data (the model) are also defined separately.

**modem**

A modulator-demodulator. A *hardware* device used to connect a digital computer to an analogue telephone network by turning analogue signals into digital signals, and vice versa.

**module**

A group of program components, typically with restricted visibility to program components in other modules. Java uses *packages* to implement this concept.

**monitor**

An object with one or more synchronized methods.

**multiple-boot options**

The hardware configurations of some computers are able to run different *operating system* and *window manager* combinations. Some systems allow a user to choose which combination they wish to use during a particular session when the computer is started, or *booted*.

**multiple inheritance**
The ability of a class or interface to extend more than one class or interface. In Java, multiple inheritance is only available in the following circumstances:

- An interface may extend more than one interface.
- A class may implement more than one interface.

Only *single inheritance* is possible for a class extending another class.

### multiprogramming system

An operating system that is able to run multiple programs concurrently.

### mutator method

A method specifically designed to allow controlled modification of a private attribute of a class. By convention, we name mutators with a *set* prefix followed by the name of the attribute being modified. For instance, the mutator for an attribute named `speed` would be `setSpeed`. By making an attribute private, we prevent objects of other classes from altering its value other than through its mutator. The mutator is able to check the value being used to modify the attribute and reject the modification if necessary. In addition, modification of one attribute might require others to be modified in order to keep the object in a consistent state. A mutator method can undertake this role. Mutators are used both to grant safe access to the value of a private attribute and to protect attributes from modification by objects of other classes. The latter goal is achieved by choosing an appropriate visibility for the mutator.

### mutual recursion

Recursion that results from two methods calling each other recursively.

### namespace

The area of a program in which particular *identifiers* are visible. Java uses *packages* to provide namespaces, and its visibility rules - private, package, protected, public - variously contain identifiers within namespaces.

### native method

A method written in a language other than Java, but accessible to a Java program. Native methods are beyond the scope of this book.

### nested class

A class defined inside an enclosing class. See *inner class*.

### newline

The 
 character.

### new operator

The operator used to create *instances* of a class.

### no-arg constructor

A constructor that takes no arguments. By default, all classes without an explicit constructor have a default no-arg constructor with public access. Its role is purely to invoke the no-arg constructor of the immediate super class.

### non-modal

A dialog is non-modal if its parent application is not blocked from further activity while the dialog is being shown. See *modal*.

### non-static nested class

See *inner class*. 
null character
The \u0000 character. Care should be taken not to confuse this with the null reference.

null reference
A value used to mean, `no object'. Used when an object reference variable is not referring to an object.

number base
The base used to interpret numerical characters. Decimal notation is base 10 and binary notation is base 2, for instance.

O

object
An instance of a particular class. In general, any number of objects may be constructed from a class definition (see singleton, however). The class to which an object belongs defines the general characteristics of all instances of that class. Within those characteristics, an object will behave according to the current state of its attributes and environment.

object construction
The creation of an object, usually via the new operator. When an object is created, an appropriate constructor from its class is invoked.

object-oriented language
Programming languages such as C++ and Java that allow the solution to a problem to be expressed in terms of objects which belong to classes.

object reference
A reference to an object. Languages other than Java use term's such as address or pointer. It is important to keep the distinction clear between an object and its reference. A variable such as argo:
Ship argo;
is capable of holding an object reference, but is not, itself, an object. It can refer to only a single object at a time, but it is able to hold different object references from time to time.

object serialization
The writing of an object's contents in such a way that its state can be restored, either at a later time, or within a different process. This can be used to store objects between runs of a program, or to transfer objects across a network, for instance.

octal
Number representation in base 8. In base 8, only the digits 0 to 7 are used. Digit positions represent successive powers of 8.

octal character constant
A character constant in the form \ddd, where each d is an octal digit. This may be used for characters with a Unicode value in the range 0-255.

operand
An operand is an argument of an operator. Expressions involve combinations of operators and operands. The value of an expression is determined by applying the operation defined by each operator to the value of its operands.

operating system
The operating system allows a computer's hardware devices to be accessed by programs. For instance, it allows data to be organized on a computer's disks in
the form of a file system and it delivers the co-ordinate positions of a mouse to programs as the mouse is moved. Operating systems also make it possible for multiple programs to be run concurrently, or multiple users to share a single machine. See concurrency.

operator
A symbol, such as -, == or ?: taking one, two or three operands and yielding a result. Operators are used in both arithmetic expressions and boolean expressions.

operator precedence
See precedence rules

out-of-bounds value
A redundant value used to indicate that a different action from the norm is required at some point. The read method of InputStream returns -1 to indicate that the end of a stream has been reached, for instance, instead of the normal positive byte-range value.

out of scope
A variable is in scope as long as the program's flow of control is within the variable's defining block. Otherwise, it is out of scope.

overriding for breadth
A form of method overriding in which the sub class version of a method implements its own behavior within the context of the attributes and behavior of the sub class and then calls the super class version so that it can perform a similar task within the super class context.

overriding for chaining
A form of method overriding in which the sub class version of a method checks to see whether it can respond to the message on its own and only calls the super class version of the method.

overriding for restriction
A form of method overriding in which the sub class version of a method calls the super class version first of all and then uses or manipulates the result or effects of that call in some way.

P

package
A named grouping of classes and interfaces that provides a package namespace. Classes, interfaces and class members without an explicit public, protected or private access modifier {access!modifier} have package visibility. Public classes and interfaces may be imported into other packages via an import statement.

package access
See package.

package declaration
A declaration used to name a package. This must be the first item in a source file, preceding any import statements. For instance,

package java.lang;

parallel programming
A style of programming in which statements are not necessarily executed in an ordered sequence but in parallel. Parallel programming languages make it easier to create programs that are designed to be run on multi-processor
hardware, for instance. Java's thread features support a degree of parallel programming.

parameter  
See argument.

parsing  
Usually applied to the action of a compiler in analyzing a program source file for syntax errors. It is also used more widely to mean the analysis of the structure of input.

pattern  
A recurring theme in class design or usage. Interfaces such as Iterator encapsulate a pattern of access to the items in a collection, while freeing the client from the need to know details of the way in which the collection is implemented.

peer  
A term used of the Abstract Windowing Toolkit (AWT) to refer to the underlying classes that provide the platform-specific implementation of component classes.

peripheral devices  
Devices attached to a computer, such as printers, disk drives, mice, etc.

pipe  
A linkage between two program components. One component acts as a source of data, and writes into the pipe. A second components acts as a receiver (sink) for the data and reads from the pipe. See PipedInputStream and PipedOutputStream.

pixel  
A 'picture element' - typically a colored dot on a screen.

polling  
The process of repeatedly testing until a condition becomes true. Polling can be inefficient if the time between tests is short compared with the time it will take for the condition to become true. A polling thread should sleep between consecutive tests in order to give other threads a chance to run. An alternative approach to polling is to arrange for an interrupt to be sent when the condition is true, or to use the wait and notify mechanism associated with threads.

polymorphism  
The ability of an object reference to be used as if it referred to an object with different forms. Polymorphism in Java results from both class inheritance and interface inheritance. The apparently different forms often result from the static type of the variable in which the reference is stored. Given the following class header

class Rectangle extends Polygon implements Comparable
an object whose dynamic type is Rectangle can behave as all of the following types: Rectangle, Polygon, Comparable, Object.

popup menu  
A menu of actions that is normally not visible on the screen until a mouse button is clicked. Popup menus help to keep a user interface from becoming cluttered.

port  
A number used by a process to communicate with another process across a network, using the Transmission Control Protocol (TCP) or User Datagram
Protocol (UDP), for instance. See TCP endpoint {Transmission Control Protocol (TCP)!endpoint}.

portable
Portability is the quality of a program that makes it possible to run it on different types of computers. Programs written in low level languages are typically not very portable because they are usually closely tied to a specific instruction set or characteristics of a particular type of Central Processing Unit. Programs written in high level languages tend to be more portable, but might still make non-portable assumptions about a computer's underlying file system, for instance. Java programs are highly portable because a lot of machine- and file-system specific details are hidden from the programmer.

post-decrement operator
See decrement operator.

post-increment operator
See increment operator.

precedence rules
The rules that determine the order of evaluation of an expression involving more than one operator. Operators of higher precedence are evaluated before those of lower precedence. For instance, in the expression $x+y*z$, the multiplication is performed before the addition because $*$ has a higher precedence than $-$. 

pre-decrement operator
See decrement operator.

preempt
The currently executing thread may be preempted, or forced to yield control, by a higher priority thread that becomes eligible to run during its timeslice.

pre-increment operator
See increment operator.

primitive type
Java's eight standard non-class types are primitive types: boolean, byte, char, double, float, int, long and short.

priority level
Each thread has a priority level, which indicates to the scheduler where it should be placed in the pecking order for being run. An eligible un-blocked thread with a particular priority will always be run before an eligible thread with a lower priority.

process
An individual thread-of-control to which an execution timeslice is allocated by an operating system.

program counter
A program counter is an integral part of a computer's Central Processing Unit. It contains a reference to the memory address of the next instruction to be fetched, ready to be executed during the next fetch-execute cycle. Immediately following an instruction fetch, the program counter is moved on to refer to the next instruction, before the current instruction is executed. The normal sequential execution of a series of instructions may be changed by executing a branch instruction, which stores a new instruction address into the program counter.

propagation
If an exception is thrown within a method, and there is no appropriate exception handler within the method, the exception may be propagated to the caller of the method. For a checked exception, the method must contain a throws clause in its header. A throws clause is not necessary for an unchecked exception to be propagated.

protected access

Protected access is available to a class member prefixed with the protected access modifier. Such a member is accessible to all classes defined within the enclosing package, and any subclasses extending the enclosing class.

protected statement

A statement within the try clause of a try statement.

protocol

A set of rules for interaction between two processes. A protocol is usually specified in a Uniform Resource Locator (URL) to indicate how a particular resource should be transferred from a Web server to the requesting client.

public interface

The members of a class prefixed with the public access modifier. All such members are visible to every class within a program.

punctuation

Symbols such as commas and semicolons, which a compiler uses to understand the structure of a program.

Q

quantum

See timeslice.

queue

See first in, first out (FIFO) queue.

quotient

When integer division is performed, the result consists of a quotient and a remainder. The quotient represents the integer number of times that the divisor divides into the dividend. For instance, in $5/3$, 5 is the dividend and 3 is the divisor. This gives a quotient of 1 and a remainder of 2.

R

radio buttons

A group of selectable components in which only one component may be selected. Selection of one of the group causes the previously selected component to be deselected.

race condition

See race hazard.

race hazard

A situation that arises between multiple threads sharing a resource. A race hazard arises when one thread's assumptions about the state of a resource are invalidated by the actions of another thread.

Random Access Memory

Random access memory, or RAM, is memory whose contents are easily accessible to the processing components of a computer. In particular, the time it takes to read and write to a particular part of the memory does not depend on

Object Oriented-Programming with Java: An Introduction
the address of the location to be read or written. This is in contrast to something like video tape which is accessed serially and, hence, the time it takes to read or write to any part of it depends on how far away the location is.

Reader class
A sub class of the Reader abstract, defined in the java.io package. Reader classes translate input from a host-dependent character set encoding into Unicode. See Writer class.

real number
A number with an integer and a fractional part. The primitive types double and float are used to represent real numbers.

recursion
Recursion results from a method being invoked when an existing call to the same method has not yet returned. For instance

```java
public static void countDown(int n){
    if(n >= 0){
        System.out.println(n);
        countDown(n-1);
    }
}
```

See direct recursion, indirect recursion and mutual recursion for the different forms this can take.

redundant value
The value of a data type that has no use or meaning within a particular context. For instance, negative values would be redundant a class using integer attributes to model assignment marks. In some applications, redundant patterns serve a useful purpose in that they can be used explicitly as out-of-bounds values or escape values.

reflection
The ability to find out what methods, fields, constructors, and so on, are defined for a class or object. Reflection is supported by the Class class, and other classes in the java.lang.reflect package. Reflection makes it possible, among other things, to create dynamic programs.

relational operators
Operators, such as <, >, <=, >=, == and !=, that produce a boolean result, as part of a boolean expression.

relative filename
A filename whose full path is relative to some point within a file system tree - often the current working folder (directory). For instance ../bin/javac.exe

A relative filename could refer to different files at different times, depending upon the context in which it is being used. See absolute filename.

repetition
See iteration.

reserved word
A word reserved for a particular purpose in Java, such as class, int, public, etc. Such words may not be used as ordinary identifiers.

resource
See Uniform Resource Locator (URL).

return statement
A statement used to terminate the execution of a method. A method with `void return type` may only have return statements of the following form:

```java
return;
```

A method with any other return type must have at least one return statement of the form:

```java
return expression;
```

where the type of `expression` must match the return type of the method.

### Return Type

The declared type of a method, appearing immediately before the method name, such as `void` in:

```java
public static void main(String[] args)
or Point[] in
```

```java
public Point[] getPoints()
```

### Return Value

The value of the `expression` used in a `return statement`.

### RGB Color Model

A color model based upon representing a color as three components: red, green and blue. See `HSB Color Model`.

### Right Shift Operator

The right shift operator (`>>`) is a `bit manipulation operator`. It moves the bits in its left operand zero or more places to the right, according to the value of its right operand. The most significant bit from before the shift is replicated in the leftmost position - this is called `sign extension`. An alternative right shift operator (`>>>`) replaces the lost bits with zeros at the left.

### Round Robin Allocation

An allocation of `timeslices` that repeatedly cycles around a set of eligible `threads` in a fixed order.

### Runtime Error

An error that causes a program to terminate when it is being run.

### Runtime Stack

A stack structure maintained by the Java Virtual Machine that records which methods are currently being executed. The most recently entered method will be at the top of the stack and the main method of an application will be near the bottom.

### S

**scheduler**

The part of the `Java Virtual Machine (JVM)` that is responsible for managing `threads`.

**scheme**

See `protocol`.

**scope**

A language's scope rules determine how widely variables, methods and classes are visible within a class or program. Local variables have a scope limited to the `block` in which they are defined, for instance. Private methods and variables have `class scope`, limiting their accessibility to their defining class. Java provides private, package, protected and public visibility.

**search path**
A list of folders (directories) to be searched - for a program or class, for instance.

security policy
A policy used to limit access by an applet to the resources of a host system.

semantic error
An error in the meaning of program. A statement may have no syntax errors, but might still break the rules of the Java language. For instance, if `ivar` is an int variable, the following statement is syntactically correct:
```
ivar = true;
```
However, it is semantically incorrect, because it is illegal to assign a boolean value to an integer variable.

server
Something that provides a service. A Web server delivers resources to its clients, for instance. When the server is an object, it is the recipient of messages from its object clients.

shallow copy
A copy of an object in which copies of all the object's sub-components are not also made. For instance, a shallow copy of an array of objects would result in two separate array objects, each containing references to the same set of objects as were stored in the original. See deep copy for an alternative.

shift operator
See left shift operator and right shift operator.

short-circuit operator
An operator in which only as many operands are evaluated as are needed to determine the final result of the operation. The logical-and (`&&`) and logical-or (`||`) operators are the most common example, although the conditional operator (`?:`) also only ever evaluates two of its three operands. See fully evaluating operator.

shortcut key
A key-press associated with a component in a Graphical User Interface (GUI) that provides an alternative to selecting the component's operation with a mouse.

sibling sub classes
Classes that have the same immediate super class. See inheritance hierarchy.

sign bit
In twos-complement notation, the most significant bit in an integer value is used to determine the sign of the value. A 1 bit indicates a negative number, and a 0 bit indicates a positive number.

sign extension
When an integer value from a type with a particular range is stored in a variable with a bigger range, Java uses sign extension to determine the resulting value. The most significant bit in the original value is used to fill the extra bits of the new value. For instance, suppose a byte variable contains the bit pattern, `10000000`. If this is stored in a short variable, the resulting bit pattern will be `111111110000000`. If the original value is `01000000`, the resulting bit pattern will be `00000001000000`.

single inheritance
In Java, a class may not extend more than one class. This means that Java has a single inheritance model for class inheritance. See multiple inheritance for the alternative.
singleton pattern
A pattern that allows us to ensure that only a single instance of a particular class exists at any one time. Such an instance is called a singleton. The pattern can also be used when instances would have no unique state and would behave identically.

software
Programs written to run on a computer.

software engineering
The system of applying an engineering discipline to the design, implementation and maintenance of software systems.

software reuse
The ability to reuse software components in different contexts. Object-oriented languages help to promote reuse by their support of encapsulation.

sound card
A hardware device used to turn digital data into sound.

stack
See last in, first out (LIFO) stack.

stack overflow
Stack overflow occurs when too many items are pushed onto a stack with a finite capacity.

stack trace
A display of the runtime stack.

state
Objects are said to possess state. The current state of an object is represented by the combined values of its attributes. Protecting the state of an object from inappropriate inspection or modification is an important aspect of class design and we recommend the use of accessor methods and mutator methods to facilitate attribute protection and integrity. The design of a class is often an attempt to model the states of objects in the real-world. Unless there is a good match between the data types available in the language and the states to be modeled, class design may be complex. An important principle in class design is to ensure that an object is never put into an inconsistent state by responding to a message.

statement
The basic building block of a Java method. There are many different types of statement in Java, for instance, the assignment statement, if statement, return statement and while loop.

statement terminator
The semicolon (;) is used to indicate the end of a statement.

static initializer
An initializer prefixed with the static reserved word. A static initializer is defined outside the methods of its enclosing class, and may only access the static fields and methods of its enclosing class.

static method
A static method (also known as a class method) is one with the static reserved word in its header. Static methods differ from all other methods in that they are not associated with any particular instance of the class to which
they belong. They are usually accessed directly via the name of the class in which they are defined.

**static nested class**
A nested class with the `static` reserved word in its header. Unlike `inner classes`, objects of static nested classes have no enclosing object. They are also known as nested top-level classes.

**static type**
The static type of an object is the declared type of the variable used to refer to it. See `dynamic type`.

**static variable**
A `static` variable defined inside a `class body`. Such a variable belongs to the class as a whole, and is, therefore, shared by all objects of the class. A class variable might be used to define the default value of an `instance variable`, for example, and would probably also be defined as `final`, too. They are also used to contain dynamic information that is shared between all instances of a class. For instance the next account number to be allocated in a bank account class. Care must be taken to ensure that access to shared information, such as this, is synchronized where multiple threads could be involved. Class variables are also used to give names to application-wide values or objects since they may be accessed directly via their containing class name rather than an instance of the class.

**stepwise refinement**
A `divide and conquer` approach to programming, in which a complex problem is recursively divided into smaller, more manageable, sub-problems. This approach to program design is often used with `structured programming` languages.

**stream class**
An input stream class is one that delivers data from its source (often the `file system` as a sequence of bytes. Similarly, an output stream class will write byte-level data. Stream classes should be contrasted with the operation of `reader` and `writer` classes.

**string**
An instance of the `String` class. Strings consist of zero or more `Unicode` characters, and they are `immutable`, once created. A literal string is written between a pair of string delimiters (""), as in "hello, world"

**structured programming**
A style of programming usually associated with languages such as C, Fortran, Pascal and so on. Using structured programming techniques, a problem is often solved using a `divide and conquer` approach such as `stepwise refinement`. An initially large problem is broken into several smaller sub-problems. Each of these is then progressively broken into even smaller sub-problems, until the level of difficulty is considered to be manageable. At the lowest level, a solution is implemented in terms of data structures and procedures. This approach is often used with `imperative programming` languages that are not `object-oriented languages`, i.e. the data structures and procedures are not implemented as classes.

**sub class**
A class that extends its super class. A sub class inherits all of the members of its super class. All Java classes are sub classes of the Object class, which is at the root of the inheritance hierarchy. See sub type

subordinate inner class
An inner class that performs well-defined subordinate tasks on behalf of its enclosing class.

sub type
A type with a parent super type. The sub-type/super-type relationship is more general than the sub-class/super-class relationship. A class that implements an interface is a sub type of the interface. An interface that extends another interface is also a sub type.

super class
A class that is extended by one or more sub classes. All Java classes have the Object class as a super class. See super type.

super type
A type with a child sub type. The sub-type/super-type relationship is more general than the sub-class/super-class relationship. An interface that is implemented by a class is a super type of the class. An interface that is extended by another interface is also a super type.

swapping
An {operating system} is often able to run programs that require more memory than is physically available on the host system. In order to do this, the full memory required is broken into smaller pieces, which are swapped in when required, and swapped out to disk when the space they occupy is required.

Swing
The Swing classes are part of a wider collection known as the Java Foundation Classes (JFC). Swing classes are defined in the javax.swing packages. They provide a further set of components that extend the capabilities of the Abstract Windowing Toolkit (AWT). Of particular significance is the greater control they provide over an application's look-and-feel.

switch statement
A selection statement in which the value of an arithmetic expression is compared for a match against different case labels. If no match is found, the optional default label is selected. For instance

```java
switch(choice){
    case 'q':
        quit();
        break;
    case 'h':
        help();
        break;
    ...
    default:
        System.out.println("Unknown command: "+choice);
        break;
}
```

swizzling
The process of recursively writing the contents of an object via object serialization.

synchronized statement
A statement in which an object-lock must be obtained for the target object before the body of the statement can be entered. Used to enclose a critical section in order to prevent a race hazard.

Syntax error
An error detected by the compiler during its parsing of a program. Syntax errors usually result from mis-ordering symbols within expressions and statements. Missing curly brackets and semicolons are common examples of syntax errors.

TCP endpoint
The combination of an IP address and Transmission Control Protocol (TCP) port number.

Ternary operator
See conditional operator

This
A Java reserved word with several different uses:

- Within a constructor, it may be used as the first statement to call another constructor in the same class. For example
  ```java
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with default values.
  public Heater()
  {
  }
  // Initialise with the given values.
  public Heater(int min, int max)
  {
      ... 
  }
  
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with the given values.
  public Heater(int min, int max)
  {
      ... 
  }
  
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with the given values.
  public Heater(int min, int max)
  {
      ... 
  }
  
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with the given values.
  public Heater(int min, int max)
  {
      ... 
  }
  
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with the given values.
  public Heater(int min, int max)
  {
      ... 
  }
  
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with the given values.
  public Heater(int min, int max)
  {
      ... 
  }
  
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with the given values.
  public Heater(int min, int max)
  {
      ... 
  }
  
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with the given values.
  public Heater(int min, int max)
  {
      ... 
  }
  
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with the given values.
  public Heater(int min, int max)
  {
      ... 
  }
  
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with the given values.
  public Heater(int min, int max)
  {
      ... 
  }
  
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with the given values.
  public Heater(int min, int max)
  {
      ... 
  }
  
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with the given values.
  public Heater(int min, int max)
  {
      ... 
  }
  
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with the given values.
  public Heater(int min, int max)
  {
      ... 
  }
  
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with default values.
  public Heater()
  {
      // Use the other constructor.
      this(15, 20);
  }
  // Initialise with the given values.
  public Heater(int min, int max)
  {
      ... 
  }
  ```

- Within a constructor or method, it may be used to distinguish between a field and a parameter or method variable of the same name. For instance:
  ```java
  public Heater(int min, int max)
  {
      this.min = min;
      this.max = max;
      ...
  }
  ```

- It can be used as a reference to the current object, typically in order to pass a reference to another object:
  ```java
  talker.talkToMe(this);
  ```

Thread
A lightweight process that is managed by the Java Virtual Machine (JVM). Support for threads is provided by the Thread class in the java.lang package.

thread starvation
A condition that applies to a thread that is prevented from running by other threads that do not yield or become blocked.

throw an exception
When an exceptional circumstance arises in a program - often as a result of a logical error, and exception object is created and thrown. If the exception is not caught by an exception handler, the program will terminate with a runtime error.

throws clause
A clause in a method header indicating that one or more exceptions will be propagated from this method. For instance
public int find(String s) throws NotFoundException

throw statement
A statement used to throw an exception. For instance
throw new IndexOutOfBoundsException(i+" is too large.");

timesharing system
An operating system that shares processor time between multiple processes by allocating each a timeslice. Once a process's timeslice has expired, another process is given a chance to run.

timeslice
The amount of running time allocated to a process or thread before the scheduler considers another to be run. A process or thread will not be able to use its full allocation of time if it becomes blocked or preempted during this period.

toggle
To alternate between two values, such as true and false, on and off, or 1 and 0.

top level class
A class defined either at the outermost level of a package or a static nested class.

Transmission Control Protocol
The Transmission Control Protocol (TCP) is a set of rules that allow reliable communication between two processes across a network. See User Datagram Protocol (UDP) for an alternative unreliable protocol.

trusted applet
An applet with more privileges than an ordinary (untrusted) applet.

try clause
See try statement.

try statement
The try statement acts as an exception handler - a place where exception objects are caught and dealt with. In its most general form, it consists of a try clause, one or more catch clauses and a finally clause.

```java
try{
    statement;
    ...
}
catch(Exception e){
    statement;
    ...
```

Object Oriented-Programming with Java: An Introduction
Either of the catch clause and finally clause may be omitted, but not both.

twos-complement notation

In twos-complement notation, the most significant bit in an integer value is used as the **sign bit**. A 1 bit indicates a negative number, and a 0 bit indicates a positive number. A positive number can be converted to its negative value by **complementing** the bit pattern and adding 1. The same operation is used to convert a negative value to its positive equivalent.

_u_

**unary operator**

An operator taking a single operand. Java's unary operators are `-`, `+`, `!`, `!`, `++` and `--`.

**unbounded repetition**

Repetition where the statements within a loop's body are performed an arbitrary number of times, according to the effects of the statements within the loop's body. All of the loop **control structures** in Java provide for unbounded repetition. See **bounded repetition**.

**unchecked exception**

An exception for which it is not required to provide a local **try statement**, or to propagate via a **throws clause** defined in the **method header**. An exception that is not handled will cause program termination if it is thrown. See **checked exception**.

**Unicode**

A 16-bit character set designed to make it easier to exchange and display information that makes use of a wide range of different languages and symbols.

**Uniform Resource Locator**

A Uniform Resource Locator (URL) extends the concept of file access from a purely local context to one in which resources are named uniformly, irrespective of where they might be physically located. A URL encodes a location (e.g. `www.javasoft.com`) a name (e.g. `index.html`) and a scheme (e.g. `http`).

**uninitialized variable**

A local variable that been declared, but has had no value assigned to it. The compiler will warn of variables which are used before being initialized.

**unnamed package**

All classes defined in files without a **package declaration** are placed in the unnamed package.

**upcast**

A cast towards an object's ultimate super type - that is, `up' the inheritance hierarchy towards the Object class, for instance

```java
// Upcast from VariableController to HeaterController
VariableController v;
...
HeaterController c = v;
```

**Object Oriented-Programming with Java: An Introduction**
See *downcast*. Java's rules of *polymorphism* mean that an explicit upcast is not usually required.

**User Datagram Protocol**

The User Datagram Protocol (UDP) is a set of rules that allow communication between two *processes* across a network. The protocol is unreliable, which means that information is not guaranteed to be transferred correctly between the two processes. See *Transmission Control Protocol (TCP)* for an alternative reliable protocol.

**UTF**

Universal Character Set (UCS) Transformation Format. A format for representing multibyte characters that is compatible with programs and *file systems* that were only designed to handle single byte characters.

**V**

**variable declaration**

The association of a variable with a particular type. It is important to make a distinction between the declaration of variables of primitive types and those of class types. A variable of primitive type acts as a container for a single value of its declared type. Declaration of a variable of a class type does not automatically cause an object of that type to be constructed and, by default, the variable will contain the value `null`. A variable of a class type acts as a holder for a reference to an object that is compatible with the variable's class type. Java's rules of *polymorphism* allow a variable of a class type to hold a reference to any object of its declared type or any of its sub types. A variable with a declared type of `Object`, therefore, may hold a reference to an object of any class, therefore.

**virtual desktop**

The name used to describe a user's graphical working area within a *window manager*. The name arose in the early days of graphical user interfaces when it was thought that these would lead to 'paperless offices'. It was anticipated that the computer screen would become a user's desktop, in which virtual documents, as opposed to paper documents, would be created, read and manipulated in various ways.

**virtual machine**

See *Java Virtual Machine (JVM)*.

**virtual memory**

A computer will have a limited amount of real memory available to it. Programs often require more memory than the amount of real memory. Furthermore, in a *multiprogramming system*, different *processes* will be competing for the same limited supply of real memory. An *operating system* overcomes these conflicts by allocating an amount of virtual memory to each process, which might be larger than the total amount of real memory. This is possible by storing unused parts of a process's *address space* on disk, until such time as it is required. When required, it is *swapped in* to part of the real memory, whose previous contents are *swapped out* to disk.

**well-known port**

A *port* number at which a *server* offers a familiar service. For instance, `80` is the well-known port number for servers using the *HyperText Transfer Protocol (HTTP)*.
W

while loop
One of Java's three control structures used for looping. The other two are the do loop and for loop. A while loop consists of a boolean expression and a loop body. The condition is tested before the loop body is entered for the first time and re-tested each time the end of the body is completed. The loop terminates when the condition gives the value false. The statements in the loop body might be executed zero or more times.

white space
Characters used to create visual spacing within a program. White spaces include space, tab, carriage return and line feed characters.

window manager
A window manager provides a computer user with a virtual desktop containing one or more windows and working areas in which individual programs may be run. Window managers allow the contents of a user's desktop to be arranged as required through resizing and arranging windows, and provide for drag-and-drop operations in collaboration with the operating system. They also monitor mouse movements to pop up menus, for instance.

wrapper classes
Java's primitive types are not object types. The wrapper classes are defined in the java.lang package. They consist of a class for each primitive type: Boolean, Byte, Character, Double, Float, Integer, Long and Short. These classes provide methods to parse strings containing primitive values, and turn primitive values into strings. The Double and Float classes also provide methods to detect special bit patterns for floating point numbers, representing values such as NaN, +infinity and -infinity.

Writer class
A sub class of the Writer abstract, defined in the java.io package. Writer classes translate output from Unicode to a host-dependent character set encoding. See Reader class.

Z

zip file
A file used to store compressed versions of files. In connection with Java bytecode files, these have largely been superseded by Java Archive (JAR) files.

Taken from the book, Object Oriented-Programming with Java: An Introduction, by David J. Barnes, published by Prentice Hall.