1.	**\"**	escape sequence
2.	**Application Program Interface (API)**	A library in Java that contains predefined classes and interfaces
3.	**Assembler**	a device used to translate assembly-language programs into machine code
4.	**Assembly Language**	uses a short descriptive word to represent each of the machine-language instructions
5.	**assignment operator**	=
6.	**assignment statement**	evaluates to the value to be assigned to a variable (=)
7.	**Augmented assignment operators**	+=, -=, **=, /= and %= (i+= 8 is i = i + 8)
8.	**Bit**	Binary digits
9.	**Block**	anything inside of a {xxxxxx}
10.	**Block Comment**	/*XXXXXXXXXXXXX*/
11.	**Boolean Expression**	An expression that evaluates a Boolean value to be true or false
12.	**boolean operators**	!, &&, ||, ^
13.	**Boolean Value**	can be true or false
14.	**Byte**	8 bits to 1 byte
15.	**Bytecode**	Similar to machine instructions, but can run on any platform with a JVM
16.	**bytecode verifier**	checks the validity of a bytecode
17.	**byte type**	-128 to 127
18.	**casting**	an operation that converts a value of one data type into a value of another data type
19.	**comment**	on a program denoted by //xxxxx or /*xxxx*/
20.	**compiler**	translates a Java source file into a Java bytecode file
21.	**concatenate strings**	using the (+) sign to combine strings
22.	**Conditional Expression (? :)**	Evaluates an expression based on a condition (pg 103)
23.	**conditional operator**	? : for if statement shorthand
24.	**constant**	a number in the program that never changes, denoted by "final"
25.	**dangling else ambiguity**	when else matches with the most recent if statement
26.	**data type**	the kind of data stored in each variable
27.	**decrement operator**	--
28.	**directive**	import
29.	**dot pitch**	the amount of space between pixels, measured in millimeters
30.	**double type**	real numbers, decimal places, twice as precise as float
31.	**escape character**	\n
32.	**expression**	represents a computation involving values, variables, and operators that, taking them together, evaluates to a value
33.	**fall-through behavior**	using no breaks in a switch
34.	**final**	denotes names
35.	**final keyword**	denotes a value as a constant
36.	**Floating point**	scientific notation
37.	**floating-point number**	Numbers with a decimal point (var double)
38.	**floating point/pi**	3.14159E1
39.	**identifier**	-names that refer to values or names - letters, digits, _, and $. -rules for creating a name in a program
40.	**identifier**	Variable
41.	**illegal identifier**	4thQtrSales
42.	**increment operator**	++
43.	**input error**	Occurs when the user inputs a value the program cannot handle
44.	**Integrated development environment**	an environment for developing Java programs
45.	**interpreter**	translates source code into machine code
46.	**int type**	an exact number, 1, 4 or 10
47.	**IPO**	input, process, output - describes simple code
48.	**Java Development Toolkit**	consists of a set of separate programs, each invoked from a command line, for developing and testing Java programs

49.	**keyword**	abstract is a
50.	**keyword**	Reserved words that have a specific meaning in java and cannot be used for variables
51.	**keywords**	do, else, and break
52.	**Literal**	Constant value directly in a program that stands for itself
53.	**literal**	a constant value that appears directly in a program
54.	**logic error**	occurs when a program does not perform the way it was intended to
55.	**long type**	a very large int, more precise
56.	**name of a type**	float
57.	**name of type**	char
58.	**name of type**	int
59.	**narrowing (of types)**	casting a data type from a large range to a smaller range - Java does this automatically
60.	**nextDouble**	method that is applied to objects of Scanner
61.	**octa integer**	075
62.	**operands**	the values operated on by a operator
63.	**operator associativity**	determine the order in which operators are evaluated
64.	**operator precedence**	var++, + and -, casting, !, * / %, + - concaction, (See page 105)
65.	**operators**	+, -, *, /, %
66.	**overflow**	If you try to store a value in a data type that cannot handle it
67.	**postdecrement**	-- placed after variable. uses original variable in expression then decreases by 1
68.	**postincrement**	++ placed after variable. uses original variable in expression then increases by 1
69.	**predecrement**	-- placed before variable. decreases variable by one, then uses it in the expression
70.	**preincrement**	++ placed before variable. increases variable by one, then uses it in the expression
71.	**preprocessor**	import statement
72.	**primitive data type**	int, real numbers, characters and booleans
73.	**Relational Operators (Boolean)**	<, <=, ==, !=, >, >=
74.	**runtime error**	errors that cause a program to terminate early, an impossible operation is detected
75.	**scope of a variable**	The part of a program where the variable can be referenced
76.	**selection statement**	statements that let you choose actions with alternative choices
77.	**short circuit operator**	same as lazy operator - && or II (and, or)
78.	**source code/program**	a high-level program's code
79.	**statement**	instructions for a high-level program
80.	**statement terminator**	;
81.	**String**	a type
82.	**syntax error**	result from errors in code construction, such as misspellings, wrong punctuation, etc.
83.	**util**	a class name in the system library that contains different java functions
84.	**variable**	represents a value stored in the computers memory
85.	**variable name**	Bool
86.	**widening (of types)**	casting from a small type to a larger type, this is done manually
87.	**wildcard import**	imports all the classes in a package by using a * **(import java.util.*;)**