

# PROGRAMMING I - JAVA

## Curriculum Content Frameworks

**Please note: All assessment questions will be taken from the knowledge portion of these frameworks.**

*Prepared by*

Marilyn Carrell, Springdale High School  
Carl Frank, Arkansas School for Mathematics, Sciences and the Arts  
Kimberly Raup, Conway High School West

*Facilitated by*

Karen Chisholm, Program Manager  
Office of Assessment and Curriculum  
Arkansas Department of Workforce Education

*Edited by*

Sandra Porter, Program Manager  
Jim Brock, Program Advisor  
Ginger Fisher, Program Advisor  
LaTrenda Jackson, Program Advisor  
Tim Johnston, Program Advisor  
Office of Business Marketing Technology  
Arkansas Department of Workforce Education

*Disseminated by*

Career and Technical Education  
Office of Assessment and Curriculum  
Arkansas Department of Workforce Education

# Curriculum Content Frameworks

## PROGRAMMING I - JAVA

Grade Levels: 9, 10, 11, 12  
Course Code: 492390

Prerequisite: Geometry or  
Algebra II

An introduction to programming and problem solving. The language used is Java. No prior programming experience is required. Good programming style is stressed. Topics included are: documentation of programs, structuring programs, program flow, decision structures, and loops.

The contents of these frameworks are not necessarily intended to be taught as independent units in this order. Many of the skills are best introduced in one unit and then spiraled back to in future units with more complexity added. However, by the end of the semester all skills should be taught.

The contents of the frameworks in Programming I, II and III are kept to the essentials. The framework team has tried to include most of the skills required in Advanced Placement Computer Science A in these three frameworks. The teacher should easily have time to include any not mentioned into the course. We believe that it is possible to teach all of the techniques in Advanced Placement Computer Science AB in three semesters. We expect that the teacher will use the remaining time in the semester to cover those topics not listed in these frameworks. Upon successful completion of Programming I – Java students are ready to enter AP Computer Science.

This course requires Java 1.5.0 or later.

### Table of Contents

	Page
Unit 1: Introduction to Programming and Ethics in Programming	1
Unit 2: Programming Techniques and Characteristics of Good Programs	2
Unit 3: Data Types and Mathematical Operations	4
Unit 4: Using Selected Standard Classes	6
Unit 5: Decision Structure	8
Unit 6: Loops	10
Glossary	12
Appendix A	18
Appendix B	22

# Unit 1: Introduction to Programming and Ethics in Programming

## Hours: 3

Terminology: Application software, Bytecode, Compiler, Hardware, High-level language, Interpreters, Low-level language, Operating system, Software, Source code, System software

CAREER and TECHNICAL SKILLS		ACADEMIC and WORKPLACE SKILLS			
What the Student Should be Able to Do		What the Instruction Should Reinforce			
Knowledge	Application	Skill Group	Skill	Description	
1.1 Define terminology	1.1.1 Prepare a list of terms with definitions	Foundation	Reading	Applies information and concepts derived from printed materials [1.3.3]  Applies/Understands technical words that pertain to JAVA Programming [1.3.6]	
1.2 Explain the difference between system and application software	1.2.1 Identify various software as either system or application	Foundations	Speaking	Comprehends ideas and concepts related to system and application software [1.2.1]	
			Reading	Analyzes and applies what has been read to system and application software [1.3.2]	
1.3 Discuss terms related to hardware and software	1.3.1 Identify technology as either hardware or software	Thinking	Knowing how to Learn	Applies new knowledge and skills to differentiate between hardware and software [4.3.1]	
1.4 Discuss various operating systems and their differences (i.e., Windows, Mac Linux)	1.4.1 Tell where each operating system is used most frequently	Foundations	Reading	Understands technical words that pertain to various operating systems and their differences [1.3.6]	
		Thinking	Reasoning	Sees relationship between various operating systems [4.5.5]	
1.5 Explain the difference between high-level and low-level languages	1.5.1 Classify commonly used programming languages as high-level or low-level	Foundations	Reading	Understands technical words that pertain to high & low level languages [1.3.6]	
		Thinking	Reasoning	Sees relationship between high and low level languages [4.5.5]	
1.6 Explain the difference between interpreters and compilers	1.6.1 Give an example of how a compiler functions vs. how an interpreter functions	Foundations	Reading	Understands technical words that compilers and interpreters [1.3.6]	
		Thinking	Reasoning	Sees relationship between interpreters and compilers [4.5.5]	
1.7 Explain the difference between executable code and bytecode	1.7.1 Draw a diagram of how a Java source code program is translated to bytecode and then to executable code	Thinking	Seeing Things in the Mind's Eye	Organizes and processes images – symbols, pictures, graphs, objects, etc. [4.6.2]	

## Unit 2: Programming Techniques and Characteristics of Good Programs

### Hours: 3

Terminology: //, Algorithm, Code block, Documentation, Logic errors, Program maintenance, Pseudocode, Run-time error, String, Syntax errors, Syntax, User-friendly, White space

CAREER and TECHNICAL SKILLS What the Student Should be Able to Do		ACADEMIC and WORKPLACE SKILLS What the Instruction Should Reinforce			
Knowledge	Application	Skill Group	Skill	Description	
2.1 Define terminology	2.1.1 Prepare a list of terms with definitions	Foundation	Reading	Applies information and concepts derived from printed materials [1.3.3]  Applies/Understands technical words that pertain to programming techniques [1.3.6]	
2.2 List the steps of the programming process	2.2.1 When given an example, be able to identify the correct step	Foundation  Thinking	Science  Writing  Reasoning	Solves practical problems using scientific methods and techniques [1.4.23]  Organizes information into an appropriate format [1.6.10]  Sees relationship between steps in the programming process [4.5.5]	
2.3 Identify the syntax of a simple program	2.3.1 Key, save, compile, and execute "Hello World" program	Foundation  Thinking	Writing  Reasoning	Organizes information into an appropriate format [1.6.10]  Applies rules and principles to a new situation [4.5.1]	
2.4 Identify the syntax to output String literals to screen	2.4.1 Write programs that use <i>System.out.println</i> with String literals	Foundation  Thinking	Writing  Reasoning	Applies rules of grammar, punctuation, capitalization, and spelling [1.6.3]  Applies rules and principles to a new situation [4.5.1]	
2.5 Identify syntax of comments	2.5.1 Use appropriate syntax to include comments in programs	Foundations  Thinking	Writing  Knowing How to Learn	Communicates thoughts, ideas, or facts in written form in a clear, concise manner [1.6.6]  Applies new knowledge and skills to properly document programs [4.3.1]	
2.6 Explain the characteristics of user-friendly programs	2.6.1 Write programs that have clear instructions  2.6.2 Write programs whose output is easy to read and understand	Foundations  Thinking	Writing  Knowing How to Learn	Communicates thoughts, ideas, or facts in written form in a clear, concise manner [1.6.6]  Applies new knowledge and skills regarding user-friendly software [4.3.1]	

CAREER and TECHNICAL SKILLS		ACADEMIC and WORKPLACE SKILLS			
What the Student Should be Able to Do		What the Instruction Should Reinforce			
Knowledge	Application	Skill Group	Skill	Description	
2.7 Explain the importance of program documentation and maintenance	2.7.1 Be able to use // and /* */ to write programs that are well-documented	Foundations	Writing	Uses technical words and concepts [1.6.4]	
	2.7.2 Update an existing program	Thinking	Problem Solving	Devises and implements a plan of action to resolve problems [4.4.3]	
2.8 Explain the importance of algorithm and/or pseudocode in program development	2.8.1 Write a psuedocode (algorithm) for a programming problem	Foundations	Reasoning	Applies rules and principles to a new situation [4.5.1]	
			Writing	Uses technical words and concepts [1.6.4]	
		Thinking	Knowing How to Learn	Communicates thoughts, ideas, or facts in written form in a clear, concise manner [1.6.6] Applies new knowledge and skills properly [4.3.1]	
2.9 Identify different types of errors (syntax, semantic, run-time, compile time)	2.9.1 When given an example, identify the error type	Foundations	Reading	Evaluates written information for accuracy, appropriateness, and style [1.3.14]	
		Thinking	Problem Solving	Identifies possible reasons for problem [4.4.6]	
2.10 Explain the characteristics of readable programs	2.10.1 Explain the characteristics of readable programs	Foundations	Reading	Identifies relevant details, facts and specifications [1.3.16]	
	2.10.2 Use descriptive identifiers		Writing	Uses language, style, organization, and format appropriate to subject matter, purpose, and audience [1.6.19]	
	2.10.3 Document difficult logic to make it easy to follow				

## Unit 3: Data Types and Mathematical Operations

### Hours: 9

**Terminology:** Boolean, Casting, Character, Concatenation, Constants, Data type, Floating point (real), Integer division, Integer, Mathematical operators, Modulus, Order of operations, Promotion (widening conversion), Round-off errors, Variable

<b>CAREER and TECHNICAL SKILLS</b>		<b>ACADEMIC and WORKPLACE SKILLS</b>			
What the Student Should be Able to Do		What the Instruction Should Reinforce			
Knowledge	Application	Skill Group	Skill	Description	
3.1 Define terminology	3.1.1 Prepare a list of terms with definitions	Foundation	Reading	Applies information and concepts derived from printed materials [1.3.3]  Applies/Understands technical words that pertain to data types and mathematical operations [1.3.6]	
3.2 List the following four primitive types: int, boolean, double, char.*  *While there are more primitive types than this, only these will be tested.	3.2.1 Compare the four data types	Foundation	Arithmetic/ Mathematics	Comprehends mathematical ideas and concepts related to the characteristics of numeric data [1.1.13]	
	3.2.2 Determine whether a particular “number” would be considered numeric		Writing	Presents answers/conclusions in a clear and understandable form [1.6.13]	
	3.2.3 Determine whether a number should be treated as an integer or a floating point (i.e. single, double)		Reasoning	Uses words appropriately [1.6.21]	
	3.2.4 Designate data type using correct syntax	Thinking	Reasoning	Comprehends ideas and concepts related to data types[4.5.2]	
3.3 Explain the use of a String object	3.3.1 Write programs that declare and utilize String objects	Foundation	Writing	Applies/Uses technical words and concepts relating to String objects [1.6.4]	
	3.3.2 Determine whether an identification number (such as Social Security Number) should be treated as a string or number	Thinking	Problem	Demonstrates logical reasoning in reaching a conclusion [4.4.2]	
3.4 Explain the purpose of concatenation	3.4.1 Write output lines using + as the concatenation operator	Foundation	Writing	Organizes information into an appropriate format [1.6.10]  Uses technical words and symbols [1.6.20]	
3.5 Explain the purpose of escape sequences (\n, \t, \\", \").	3.5.1 Write programs that use escape sequences to print strings	Foundations	Writing	Organizes information into an appropriate format [1.6.10]  Uses technical words and symbols [1.6.20]	

CAREER and TECHNICAL SKILLS		ACADEMIC and WORKPLACE SKILLS			
What the Student Should be Able to Do		What the Instruction Should Reinforce			
Knowledge	Application	Skill Group	Skill	Description	
3.6 Explain the advantages of using integer variables whenever possible (faster computation, require less memory, obtain exact answers)	3.6.1 Use integer variables in programs where appropriate	Foundations Thinking	Arithmetic/ Mathematics  Decision Making	Applies mathematical principles related to integer variables. [1.1.4]  Comprehends ideas and concepts related to the characteristics of numeric data [4.2.2]	
3.7 Explain the advantages and disadvantages of floating-point numbers (round-off errors, more memory, approximate answers, slower computation, size of numbers to be stored, etc.)	3.7.1 Use floating point variables in programs where appropriate	Foundations Thinking	Arithmetic/ Mathematics  Writing  Reasoning	Applies mathematical principles related to floating-point variables [1.1.4]  Uses words appropriately [1.6.21]  Comprehends ideas and concepts related to floating point variables [4.5.2]	
3.8 List arithmetic operations and order of operations (*, /, %, +, -)	3.8.1 Write formulas using operators and order of operations  3.8.2 Write programs that use mathematical operations correctly (integer arithmetic vs floating point arithmetic)	Foundations Thinking	Arithmetic/ Mathematics  Decision Making	Comprehends mathematical ideas and concepts related to the characteristics of arithmetic operations and order of operations [1.1.13]  Comprehends ideas and concepts related to writing formulas [4.2.2]	
3.9 Explain the difference in promotion and casting.	3.9.1 Determine the correct answer to math expressions where casting and promotion are involved  3.9.2 Write math expressions correctly that use promotion and casting	Foundations	Arithmetic/ Mathematics	Calculate mathematic expression as it pertains to casting and promotion [1.1.8]  Expresses mathematical ideas and concepts orally and in writing [1.1.23]	
3.10 Explain rules for choosing variable names	3.10.1 Write programs that use descriptive variable names  3.10.2 Write programs that use descriptive variable names	Foundation Thinking	Writing  Learning  Reasoning	Uses words appropriately [1.6.21]  Applies new knowledge and skills to choosing variable names [4.3.1]  Comprehends ideas and concepts related to programs with appropriate variable names [4.5.2]	
3.11 Explain the circumstances and give examples of appropriate occasions to use constants.	3.11.1 Use constants when appropriate in programs	Foundation Thinking	Writing  Reasoning	Uses words appropriately [1.6.21]  Comprehends ideas and concepts related to using constants [4.5.2]	

## Unit 4: Using Selected Standard Classes

### Hours: 9

Note to Teachers: The Scanner class from Java 1.5.0 and later is used for keyboard input. There is an explanation in the supplementary materials following the glossary. The random method is from the Math class.

Terminology: Arguments, Class, Interactive program, Method, Object, Parameters (formal parameters), Prompt, Pseudorandom, Return type

CAREER and TECHNICAL SKILLS		ACADEMIC and WORKPLACE SKILLS			
What the Student Should be Able to Do		What the Instruction Should Reinforce			
Knowledge	Application	Skill Group	Skill	Description	
4.1 Define terminology	4.1.1 Prepare a list of terms with definitions	Foundation	Reading	Applies information and concepts derived from printed materials [1.3.3]  Applies/Understands technical words that pertain to standard classes [1.3.6]	
4.2 Explain the purpose of the compiler directive <i>import</i>	4.2.1 Write programs that import the selected classes	Foundation  Thinking	Writing  Reasoning	Uses languages, style, organization, and format appropriate to subject matter, purpose, and audience [1.6.19]  Comprehends ideas and concepts related to formatting [4.5.2]	
4.3 Give examples of meaningful prompts	4.3.1 Write programs with meaningful prompts	Foundations  Thinking	Writing  Knowing How to Learn	Communicates thoughts, ideas, or facts in written form in a clear, concise manner [1.6.6]  Applies new knowledge and skills to properly prompt user(s) of the program [4.3.1]	
4.4 Describe how to instantiate a Scanner object. <i>(java.util.Scanner)</i>	4.4.1 Write a program that instantiates a Scanner object	Foundations  Thinking	Writing  Knowing how to learn	Organizes information into an appropriate format [1.6.10]  Applies new knowledge and skills to allow input from the user(s) of the program [4.3.1]	
4.5 Describe how to read the following data types from the keyboard using the Scanner object: <code>nextLine()</code> , <code>nextInt()</code> , <code>nextDouble()</code> <i>(java.util.Scanner)</i>	4.5.1 Expand the program above so that it reads Strings, ints and doubles from the keyboard	Foundations  Thinking	Writing  Knowing How to	Applies/Uses technical words and concepts [1.6.4]  Applies new knowledge and skills to allow input from the user(s) of the program [4.3.1]	
4.6 Describe how to close a Scanner object <i>(java.util.Scanner)</i>	4.6.1 Revise the program above so that it closes the scanner object	Foundations  Thinking	Writing  Knowing How to Learn	Applies/Uses technical words and concepts [1.6.4]  Applies new knowledge and skills closing a scanner object [4.3.1]	

CAREER and TECHNICAL SKILLS		ACADEMIC and WORKPLACE SKILLS			
What the Student Should be Able to Do		What the Instruction Should Reinforce			
Knowledge	Application	Skill Group	Skill	Description	
4.7 Describe how to set up a variable to print currency or percents using NumberFormat and how to use the <i>format</i> method ( <i>java.text.NumberFormat</i> )	4.7.1 Write a program that displays the output in currency and percent formats	Foundations	Writing	Applies/Uses technical words and concepts [1.6.4]  Organizes information into an appropriate format [1.6.10]	
4.8 Describe how to instantiate a DecimalFormat object and how to use # and 0 to print the specified digits ( <i>java.text.DecimalFormat</i> )	4.8.1 Write a program that instantiates a DecimalFormat object	Foundations	Writing	Applies/Uses technical words and concepts [1.6.4]  Organizes information into an appropriate format [1.6.10]	
4.9 Describe how to print using a specific number of digits before and/or after the decimal place with the <i>format</i> method from DecimalFormat ( <i>java.text.DecimalFormat</i> )	4.9.1 Write a program that displays the output with a specified number of decimal places	Foundations  Thinking	Writing  Knowing How to Learn	Organizes information into an appropriate format [1.6.10]  Applies new knowledge and skills to display output in a specific format [4.3.1]	
4.10 Describe how to use the Math class to get a square root, power and/or absolute value ( <i>sqrt</i> , <i>pow</i> , <i>abs</i> methods)	4.10.1 Write a program that uses <i>sqrt</i> , <i>pow</i> , and <i>abs</i> Math functions in calculations	Foundations	Arithmetic/ Mathematics	Applies mathematical principles related to Math functions [1.1.4]  Comprehends a mathematical ideas and concepts related to Math.Random [1.1.13]	
4.11 Describe how to obtain a random double and explain the range of possible values ( <i>Math class</i> )	4.11.1 Write a program that obtains and uses random numbers using <i>Math.Random()</i>	Foundations  Thinking	Arithmetic/ Mathematics  Problem Solving	Applies mathematical principles related to Math.Random functions [1.1.4]  Comprehends ideas and concepts related to Math.Random functions [4.4.1]	
4.12 Describe how to obtain an integer random integer in the range of 0 .. N or 1 .. N inclusive ( <i>Math class</i> )	4.12.1 Write a program that obtains and uses random integers in the ranges 1 .. N and 0 .. N using ( <i>int</i> ) $((high - low + 1) * Math.Random() + low)$ formula	Foundations  Thinking	Arithmetic/ Mathematics  Problem Solving	Applies mathematical principles related to Math.Random functions [1.1.4]  Comprehends ideas and concepts related to integer random integer [4.4.1]	

## Unit 5: Decision Structure

### Hours: 15

Terminology: Boolean expression, Logical operators, Nested decision statements, Relational operator, Short-circuit, Truth tables

CAREER and TECHNICAL SKILLS What the Student Should be Able to Do		ACADEMIC and WORKPLACE SKILLS What the Instruction Should Reinforce			
Knowledge	Application	Skill Group	Skill	Description	
5.1 Define terminology	5.1.1 Prepare a list of terms with definitions	Foundation	Reading	Applies information and concepts derived from printed materials [1.3.3]  Applies/Understands technical words that pertain to decision structure [1.3.6]	
5.2 List relational operators	5.2.1 Write boolean expressions that use the appropriate relational operator	Foundation  Thinking	Arithmetic/ Mathematics  Reasoning	Interprets mathematical symbols [1.1.26]  Comprehends ideas and concepts related to relational operators [4.5.2]	
5.3 Describe the process of comparing two strings	5.3.1 When given two strings, determine if they are equal, the first is smaller, or the first is larger	Foundation  Thinking	Arithmetic/ Mathematics  Learning  Reasoning	Interprets mathematical symbols [1.1.26]  Applies new knowledge and skills to strings [4.3.1]  Comprehends ideas and concepts related to the comparison of two strings [4.5.2]	
5.4 Explain how to use the <i>compareTo()</i> and <i>equals()</i> methods from the String class	5.4.1 Write boolean expressions that compare strings for equality and order	Foundation  Thinking	Arithmetic/ Mathematics  Learning How to Learn	Interprets mathematical symbols [1.1.26]  Applies new knowledge and skills appropriately [4.3.1]	
5.5 Explain the syntax and logic of <i>if</i> statements	5.5.1 Write programs that use <i>if</i> statements  5.5.2 Write programs that use braces correctly to form block <i>if</i> statements	Foundation  Thinking	Writing  Reasoning	Applies rules of punctuation [1.6.3]  Organizes information into an appropriate format [1.6.10]  Comprehends ideas and concepts related to the logic of an <i>if</i> statement [4.5.2]	
5.6 Explain the syntax and logic of <i>if-else</i> statements	5.6.1 Write statements that use <i>if-else</i> to make the correct decision based on the data; use braces where needed to block the statements	Foundation  Thinking	Writing  Reasoning	Organizes information into an appropriate format [1.6.10]  Comprehends ideas and concepts related to the logic of an <i>if-else</i> statement [4.5.2]	

CAREER and TECHNICAL SKILLS What the Student Should be Able to Do		ACADEMIC and WORKPLACE SKILLS What the Instruction Should Reinforce			
Knowledge	Application	Skill Group	Skill	Description	
5.7 Explain the syntax and logic of nested statements	5.7.1 Write programs using block <i>if-else</i> for 3 or more alternatives	Foundation  Thinking	Writing  Reasoning	Organizes information into an appropriate format [1.6.10]  Comprehends ideas and concepts related to the logic of an <i>if-else</i> statement [4.5.2]	
5.8 Explain the use of logical operators <i>and</i> , <i>or</i> , and <i>not</i> (&&,   , !)	5.8.1 Write programs which require the use of &&,   , and !  5.8.2 Write a program that requires the use of short-circuit and (&&) and short-circuit or (  )	Foundation   Thinking	Arithmetic/ Mathematics  Writing  Problem Solving	Applies mathematical principles related to logical operators [1.1.4]  Organizes information into an appropriate format [1.6.10]  Demonstrates logical reasoning in reaching a conclusion [4.4.2]	

## Unit 6: Loops

### Hours: 21

Terminology: Accumulators, Counters, Entrance condition loops, Exit condition loops, Nested loops

CAREER and TECHNICAL SKILLS What the Student Should be Able to Do		ACADEMIC and WORKPLACE SKILLS What the Instruction Should Reinforce			
Knowledge	Application	Skill Group	Skill	Description	
6.1 Define terminology	6.1.1 Prepare a list of terms with definitions	Foundation	Reading	Applies information and concepts derived from printed materials [1.3.3]  Applies/Understands technical words that pertain to loops [1.3.6]	
6.2 Describe the purpose and syntax of <i>for</i> loops	6.2.1 Write programs that use <i>for</i> loops	Foundation  Thinking	Writing  Reasoning	Organizes information into an appropriate format [1.6.10]  Comprehends ideas and concepts related to the logic of an <i>for</i> loop [4.5.2]	
6.3 Explain the procedure to use <i>for</i> loops to count in increments/ decrements other than one	6.3.1 Write counting <i>for</i> loops with increments other than 1	Foundation  Thinking	Arithmetic  Writing  Reasoning	Applies mathematical principles related to <i>for</i> loops and increments [1.1.4]  Organizes information into an appropriate format [1.6.10]  Comprehends ideas and concepts related to the logic of an <i>for</i> loop [4.5.2]	
6.4 Explain the syntax of nested loops	6.4.1 Determine the output of a nested loop  6.4.2 Write programs that use nested <i>for</i> loops	Foundation  Thinking	Writing  Problem Solving	Organizes information into an appropriate format [1.6.10]  Demonstrates logical reasoning in reaching a conclusion [4.4.2]	
6.5 Explain the logic of <i>while</i> loops	6.5.1 Write programs that use <i>while</i> loops	Foundation  Thinking	Writing  Reasoning	Organizes information into an appropriate format [1.6.10]  Comprehends ideas and concepts related to the logic of an <i>for</i> loop [4.5.2]	

CAREER and TECHNICAL SKILLS What the Student Should be Able to Do		ACADEMIC and WORKPLACE SKILLS What the Instruction Should Reinforce			
Knowledge	Application	Skill Group	Skill	Description	
6.6 Explain the process of using counters with loops	6.6.1 Write programs that use counters with loops	Foundation  Thinking	Arithmetic/ Mathematics  Writing  Reasoning	Applies mathematical principles related to using loops with counters [1.1.4]  Organizes information into an appropriate format [1.6.10]  Comprehends ideas and concepts related to counters with loops[4.5.2]	
6.7 Explain the logic of using accumulators with loops	6.7.1 Write programs that use accumulators with loops	Foundation  Thinking	Arithmetic  Writing  Reasoning	Applies mathematical principles related to using accumulators with loops [1.1.4]  Organizes information into an appropriate format [1.6.10]  Comprehends ideas and concepts related to using accumulators with loops [4.5.2]	
6.8 Explain the difference in the effect of a <i>while</i> loop (entrance condition loop) and a <i>do while</i> (exit condition loop) loop	6.8.1 Write programs that use <i>do while</i> loops	Foundation  Thinking	Arithmetic/ Mathematics  Writing  Reasoning	Applies mathematical principles related to <i>while</i> and <i>do while</i> loops [1.1.4]  Organizes information into an appropriate format [1.6.10]  Comprehends ideas and concepts related to the logic of <i>while</i> and <i>do while</i> loops [4.5.2]	

## Glossary

### Unit 1: Introduction to Programming and Ethics in Programming

1. Application software – programs that perform a specific task, such as games, word processing, and spreadsheets
2. Bytecode – the result of the compiler which must still be interpreted using the Java Virtual Machine; bytecode is stored in class files
3. Compiler – a program that converts the entire program into machine code for most languages; in Java, it converts to bytecode
4. Hardware – the physical components of the computer
5. High-level language – programming language that is made up of English-like instructions
6. Interpreters – a program that translates and executes code; also known as the Java Virtual Machine (JVM) and it "interprets" bytecode
7. Low-level language – machine language (first-generation) and assembly language (second generation)
8. Operating system – computer software that runs the computer – it allows the user to communicate, save, print, load programs, etc
9. Software – computer instructions, also called programs or applications
10. Source code – the program written in its original language, such as Java
11. System software – programs that run the computer and its components

## Unit 2: Programming Techniques and Characteristics of Good Programs

1. `//` – escape characters that begin a line of comment, which is ignored by the compiler
2. Algorithm – step-by-step process for solving a problem
3. Code block – a group of one or more statements enclosed within opening and closing braces
4. Documentation – information about a program, which includes comments inside the program as well as user's guides
5. Logic errors – a problem caused by incorrect coding which produces incorrect results, rather than causing the computer to crash
6. Program maintenance – the process of fixing errors in and updating software
7. Pseudocode – algorithm written in a combination of English and programming code ("fake or pretend program")
8. Run-time error – a problem that occurs as the program is executing, which causes it to crash (such as a division by zero error)
9. String – a series of characters; in Java, they are represented by String objects and string literals such as "hello"
10. Syntax errors – an error caused by breaking a language's grammar rules
11. Syntax – the grammar rules of a programming language
12. User-friendly – a program that is easy for the user to understand and use
13. White space – the insertion of tabs and line spacing that allows the program to be more readable

## Unit 3: Data Types and Mathematical Operations

1. Boolean – a data type that can only take the values of true or false
2. Casting – used to convert a value of one type to a value of a different type; i.e. double to int
3. Character – a single letter, symbol, digit, or punctuation mark represented in Unicode
4. Concatenation – the process of attaching the end of one string to the beginning of another string, producing a longer string
5. Constants – a named memory cell that contains a value that cannot be changed from its initial value
6. Data type – a description of a set of values that a variable can have; i.e. int, double, char, and Boolean
7. Floating point (real) – a number that has a fractional component; numbers that contain decimal portions
8. Integer division – a type of division performed on an integer which returns the quotient
9. Integer – a positive or negative whole number or zero
10. Mathematical operators – symbols that represent mathematical operations: +, -, \*, /, %, ++, --, +=, -=, \*=, /=, %=
11. Modulus – a type of division performed by the Mod operator which returns the remainder portion
12. Order of operations – the order that is used to perform mathematical calculations (parentheses, exponents, multiplication/division, addition/subtraction)
13. Promotion (widening conversion) – automatic conversion of compatible data types when the resulting field is larger than the source; i.e. double x = 2 \* 2 result would yield a value of 4.0 rather than int 4
14. Round-off errors – a round-off error occurs when a floating-point value cannot be stored in the allotted space and is rounded off
15. Variable – a named memory location that stores a value

## Unit 4: Using Selected Standard Classes

1. Arguments (actual parameters) – a value or expression passed in a method call
2. Class – a description of the attributes and behavior of a set of computational objects
3. Interactive program – a program that requires user input when executing
4. Method – a named set of code that can be treated as a unit and invoked by name
5. Object – an encapsulated collection of data variables and methods, an instance of a class
6. Parameters (formal parameters) – a parameter name in a method receiving its value from the actual parameter passed to it
7. Prompt – a message that asks the user for information
8. Pseudorandom – a number that a program seems to create at random but that really comes from a seed value
9. Return type – a type of value returned by a method

## Unit 5: Decision Structure

1. Boolean expression – an expression that gives a true or false result, mostly used in selection and repetition statements
2. Logical operators – one of the operators that perform a logical NOT (!), AND (&&), or OR (||), returning a Boolean result
3. Nested decision statements – a decision statement used within another decision statement
4. Relational operator – an operator used for comparison of data items of the same type
5. Short-circuit – when using logical operators in a Boolean expression, if the left operand can determine the result, the right operand is not evaluated
6. Truth tables – a complete list of all of the possible values in a Boolean expression

## Unit 6: Loops

1. Accumulators – a variable used for the purpose of summing successive values of some other variable
2. Counters – a variable used to count the number of times some process is completed
3. Entrance condition loops – a loop that tests the condition before going into the loop body, will execute 0 or more times (*for* and *while* loops)
4. Exit condition loops – a loop that tests the condition at the bottom of the loop; therefore, will always execute at least one time (*do while* loop)
5. Nested loops – a loop as one of the statements in the body of another loop

# Appendix A

## Using the Scanner Class

Students should use the Scanner class to read data from the keyboard. To do that, the student must import the scanner class. (See example below.)

**Step 1:** import java.util.Scanner;

**In the main method:**

**Step 2:** instantiate a Scanner object;

**Step 3:** read the data using the following methods—

next() Returns a string from the input stream

*Note: It will skip over white space. However, the delimiter for the string is white space (space, tab, enter). Therefore, it will read only the first word of a multiple word response.*

■ nextLine() returns the string **up to the end of line character** from the input stream.

*Note: This method accepts multiple word responses.*

■ nextDouble() returns the double read from the input stream

■ nextInt() returns the int read from the input stream

■ nextBoolean() returns the boolean read from the input stream

**Step 4:** close the input stream using the close() method.

**Example of a program that uses only numbers:**

```
import java.util.Scanner;
```

```
public class ScannerPractice
```

```
{  
    public static void main(String[] args)  
    {  
        Scanner input = new Scanner(System.in);  
  
        String name = "";  
        int numStudents = 0;  
        double avg = 0.0;  
  
        System.out.print ("Enter the number of students: ");  
        numStudents = input.nextInt();  
  
        for (int x = 1; x <= numStudents; x++)
```

```

        {
            System.out.print("Enter the average for Student " + x + ": ");
            avg = input.nextDouble();
            if (avg>=59.5)
            {
                System.out.println("Student " + x + ": Passing");
            }
            else
            {
                System.out.println("Student " + x + ": Failing");
            }
        }
    }
    input.close();
}

```

**Example of a very simple program that reads Strings using the readLine() method and numbers:** Generally, students will encounter no problems reading Strings as long as they read the strings first and then read the numbers.

```

import java.util.Scanner;

public class ScannerPractice
{
    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);

        System.out.print ("Enter your name: ");
        String name = input.nextLine();
        //The nextLine() method reads everything up to the end of line
        System.out.print ("Enter your age: ");
        int age = input.nextInt();
        System.out.println (name + ", you sure are old. You are "
            + (age * 365 * 24 * 60 *60) + " seconds old.");

        input.close();
    }
}

```

**Example of using Strings with the next() method.** The problem with next is that it reads to the white space; thus, using only the first name and trying to make the middle name the age (which throws an exception).

```

public class ScannerPractice
{
    public static void main(String[] args)
    {

```

```

Scanner input = new Scanner(System.in);

System.out.print ("Enter your name: ");
String name = input.next();
//If the student tries to enter Mary Ann as the name, Mary becomes
//the name and then it attempts to use Ann (the next data in the
//stream) as the age. This will throw an exception.
System.out.print ("Enter your age: ");
int age = input.nextInt();
System.out.println (name + ", you sure are old. You are "

input.close();
}
}

```

**Example of using the readLine() method following either readInt() or readDouble:** Once an int or a double has been read, the end-of-line character is left in the stream. Therefore, the readLine() will pick up the end of line character as the data for the next String variable rather than the data on the following line. Here is how to avoid that.

```
import java.util.Scanner;
```

```

public class ScannerPractice
{
    public static void main(String[] args)
    {
        Scanner

        String
        int
        double

        System.out.print ("Enter the number of students: ");
        numStudents = input.nextInt();
        endOfLineCharacter = input.nextLine();
        //reads the end of line character into a "junk" variable, which
        //advances the input stream to the next real data and prepares for
        //the following nextLine();
        for (int x = 1; x <= numStudents; x++)
        {
            System.out.print("Enter the student's name: ");
            name = input.nextLine();
            System.out.print("Enter the average for " +name + ": ");
            avg = input.nextDouble();
            endOfLineCharacter = input.nextLine();
            //reads the end of line character into a "junk" variable, which
            //advances the input stream to the next real data and prepares
            //for the following nextLine(); when it loops back.

```

```
        if (avg>=59.5)
        {
            System.out.println(name + ": Passing");
        }
        else
        {
            System.out.println(name + ": Failing");
        }
    }
    input.close();
}
```

## Appendix B

### Using the random() method from the Math class

When your students are tested they will be tested using the random() method from the Math class—not the random method from the util class. Nothing has to be imported. There is no “randomize” command like in many other languages.

The random() method from the Math class returns number in the range of  $0 \leq \text{num} < 1$ :

```
double num = Math.random();
```

However, most of the time, one needs an integer within a range. The generic formula is:

```
int num = (int) ( (high – low + 1) * Math.random() + low);
```

To get a number between 1 and 100, the formula is

```
int num = (int) ( (100-1+1) * Math.random() + 1); -or-  
int num = (int) (100 * Math.random() + 1);
```

To get a number between 0 and 100, the formula is

```
int num = (int) ( (100-0+1) * Math.random() + 0); -or-  
int num = (int) (101 * Math.random());
```

To get a number between 50 and 75, the formula is

```
int num = (int) ( (75 – 50 + 1) * Math.random() + 75); -or-  
int num = (int) (26 * Math.random() + 75);
```