APPENDIX A

AP Computer Science Java Subset

The AP Java subset is intended to outline the features of Java that may appear on the AP Computer Science A Exam. The AP Java subset is NOT intended as an overall prescription for computer science courses — the subset itself will need to be supplemented in order to address all topics in a typical introductory curriculum. For example, input and output must be part of a course on computer programming. However, there are many ways to handle input and output in Java. Because of this variation, details of input and output (except for basic text output using `System.out.print` and `System.out.println`) are not tested on the AP Computer Science A Exam.

This appendix describes the Java subset that students will be expected to understand when they take the AP Computer Science A Exam. A number of features are also mentioned that are potentially relevant in an introductory computer science course but are not tested on the exam. Omission of a feature from the AP Java subset does not imply any judgment that the feature is inferior or not worthwhile.

The AP Java subset was selected to

1. enable the test designers to formulate meaningful questions.
2. help students with test preparation.
3. enable instructors to follow a variety of approaches in their courses.

To help students with test preparation, the AP Java subset was intentionally kept small. Language constructs and library features were omitted that did not add significant functionality and that can, for the formulation of exam questions, be expressed by other mechanisms in the subset.

The AP Java subset gives instructors flexibility in how they use Java in their course. For example, some courses teach how to perform input/output using streams or readers/writers, others teach graphical user interface construction, and yet others rely on a tool or library that handles input/output. For the purpose of the AP Computer Science A Exam, these choices are incidental and are not central for the problem solving process or for the mastery of computer science concepts. The AP Java subset does not address handling of user input at all. That means that the subset is not complete. To create actual programs, instructors need to present additional mechanisms in their courses.

The following section contains the language features that may be tested on the AP Computer Science A Exam. The Java Quick Reference contains a list specifying which Standard Java classes, interfaces, constants, and methods may be used on the exam. This document is available to students when they take the exam, is available at AP Central, and is included in Appendix B.

Language Features and other Testable Topics

Tested in the AP CS A Exam	Notes	Not tested in the AP CS A Exam, but potentially relevant/useful
Comments `/* */`, `//`, and `/** */` Javadoc `@param` and `@return` comment tags		Javadoc tool
Primitive Types `int`, `double`, `boolean`		`char`, `byte`, `short`, `long`, `float`

Operators
Arithmetic: `+`, `-`, `*`, `/`, `%`
Increment/Decrement: `++`, `--`
Assignment: `=`, `+=`, `-=`, `*=`, `/=`, `%=`
Relational: `==`, `!=`, `<`, `<=`, `>`, `>=`
Logical: `!`, `&&`, `||`
Numeric casts: `(int)`, `(double)`
String concatenation: `+`
	1, 2, 3, 4, 5	`&`, `|`, `^`
		`(char)`, `(float)`
		`StringBuilder`
		Shift: `<<`, `>>`, `>>>`
		Bitwise: `~`, `&`, `|`, `^`
		Conditional: `?:`

Object Comparison object identity `(==, !=)` vs. object equality `(equals)`, `String compareTo`		implementation of `equals` `Comparable`
Escape Sequences `\"`, `\\`, `\n` inside strings		`\'`, `\t`, `\unnnn`
Input / Output `System.out.print`, `System.out.println`	6	`Scanner`, `System.in`, `System.out`, `System.err`, `Stream` input/output, GUI input/output, parsing input: `Integer.parseInt`, `Double.parseDouble` formatting output: `System.out.printf`

Tested in the AP CS A Exam	Notes	Not tested in the AP CS A Exam, but potentially relevant/useful

Exceptions
`ArithmeticException`,
`NullPointerException`,
`IndexOutOfBoundsException`,
`ArrayIndexOutOfBoundsException`,
`IllegalArgumentException`
		`try/catch/finally`
		`throw, throws`
		`assert`

Arrays 1-dimensional arrays, 2-dimensional rectangular arrays, initializer list: `{ ... }`, row-major order of 2-dimensional array elements	7, 8	`new` *type*`[]{ ... }`, ragged arrays (non-rectangular), arrays with 3 or more dimensions

Control Statements
`if, if/else,`
`while, for,`
enhanced `for` (for-each),
`return`
		`switch,`
		`break, continue,`
		`do-while`

Variables parameter variables, local variables, `private` instance variables: visibility `(private)` `static` (class) variables: visibility `(public, private)`, `final`		`final` parameter variables, `final` local variables, `final` instance variables

Methods
visibility `(public, private)`,
`static, non-static,`
method signatures,
overloading, overriding,
parameter passing
	9, 10	visibility `(protected)`,
		`public static void`
		`main(String[] args)`,
		command line arguments,
		variable number of parameters,
		`final`

Constructors `super(), super(`*args*`)`	11, 12	default initialization of instance variables, initialization blocks, `this(`*args*`)`

Tested in the AP CS A Exam	Notes	Not tested in the AP CS A Exam, but potentially relevant/useful
Classes `new`, visibility `(public)`, accessor methods, modifier (mutator) methods Design/create/modify class. Create subclass of a superclass `(abstract, non-abstract)`. Create class that implements an interface.	13, 14	`final`, visibility `(private, protected)`, nested classes, inner classes, enumerations
Interfaces Design/create/modify an interface.	13, 14	
Inheritance Understand inheritance hierarchies. Design/create/modify subclasses. Design/create/modify classes that implement interfaces.		
Packages `import` *packageName.className*		`import` *packageName*`.*` , `static` import, `package` *packageName* , class path
Miscellaneous OOP "is-a" and "has-a" relationships, `null`, `this`, `super.`*method(args)*	15, 16	`instanceof` *(class)* cast `this.`*var*, `this.`*method*`(`*args*`)`,
Standard Java Library `Object`, `Integer, Double`, `String`, `Math`, `List<E>, ArrayList<E>`	17, 18	`clone`, autoboxing, `Collection<E>`, `Arrays, Collections`