Language Features

1. The primitive types `int`, `double`, and `boolean` are part of the AP Java subset. The other primitive types `short`, `long`, `byte`, `char`, and `float` are not in the subset. In particular, students need not be aware that strings are composed of `char` values. Introducing `char` does not increase the expressiveness of the subset. Students already need to understand string concatenation, `String.substring`, and `String.equals`. Not introducing `char` avoids complexities with the char/int conversions and confusion between `"x"` and `'x'`.
2. Arithmetic operators: `+`, `-`, `*`, `/`, `%` are part of the AP Java subset.
3. The increment/decrement operators `++` and `--` are part of the AP Java subset. These operators are used only for their side effect, not for their value. That is, the postfix form (for example, `x++`) is always used, and the operators are not used inside other expressions. For example, `a[x++]` is not used.
4. The assignment operator `=` is part of the AP Java subset. The combined arithmetic/assignment operators `+=`, `-=`, `*=`, `/=`, `%=` are part of the AP Java subset, although they are used simply as a shorthand and will not be used in the adjustment part of a `for` loop.
5. Relational operators `==`, `!=`, `<`, `<=`, `>`, `>=` are part of the AP Java subset.
6. Logical operations `&&`, `||`, `!` are part of the AP Java subset. Students need to understand the "short circuit" evaluation of the `&&` and `||` operators. The logical `&`, `|` and `^` and the bit operators `<<`, `>>`, `>>>`, `&`, `~`, `|`, `^` are not in the subset.
7. The ternary `?:` operator is not in the subset.
8. The numeric casts `(int)` and `(double)` are part of the AP Java subset. Since the only primitive types in the subset are `int`, `double`, and `boolean`, the only required numeric casts are the cast `(int)` and the cast `(double)`. Students are expected to understand "truncation towards 0" behavior as well as the fact that positive floating-point numbers can be rounded to the nearest integer as `(int)(x + 0.5)`, negative numbers as `(int)(x - 0.5)`. Autoboxing, that is, the automatic conversion between primitive types `(int, double)` and the corresponding wrapper classes `(Integer, Double)` is not in the subset.
9. String concatenation `+` is part of the AP Java subset. Students are expected to know that concatenation converts numbers to strings and invokes `toString` on objects. String concatenation can be less efficient than using the `StringBuffer` class. However, for greater simplicity and conceptual clarity, the `StringBuffer` class is not in the subset.
10. The escape sequences inside strings `\\`, `\"`, `\n` are part of the AP Java subset. The `\t` escape and Unicode `\uxxxx` escapes are not in the subset. The `\'` escape is only necessary inside character literals and is not in the subset.

11. User input is not part of the AP Java subset. There are many possible ways for supplying user input: e.g., by reading from a `Scanner`, reading from a stream (such as a file or a URL), or from a dialog box. There are advantages and disadvantages to the various approaches. The exam does not prescribe any one approach. Instead, if reading input is necessary, it will be indicated in a way similar to the following:

```
double x = /* call to a method that
      reads a floating-point number */;
```

or

```
double x = . . .;
      // read user input
```

Converting strings to numeric values (e.g., with `Integer.parseInt`) is not in the subset.

12. Testing of output is restricted to `System.out.print` and `System.out.println`. As with user input, there are many possible ways for directing the output of a program: e.g., to `System.out`, to a file, or to a text area in a graphical user interface. The AP Java subset includes the ability to print output to `System.out`, because it makes it easy to formulate questions. Since most graphical environments allow printing of debug messages to `System.out` (with output being collected in a special window, e.g., the "Java console" in a browser), students are usually familiar with this method of producing output. Formatted output (e.g., with `NumberFormat` or `System.out.printf`) is not in the subset.
13. The `main` method and command-line arguments are not in the subset. The AP Computer Science Development Committee does not prescribe any particular approach for program invocation. In free-response questions, students are not expected to invoke programs. In case studies, program invocation with `main` may occur, but the `main` method will be kept very simple.
14. Arrays: One-dimensional arrays and two-dimensional rectangular arrays are part of the AP Java subset. Both arrays of primitive types (e.g., `int[]`) and arrays of objects (e.g., `Student[]`) are in the subset. Initialization of named arrays (`int[] arr = { 1, 2, 3 };`) is part of the AP Java subset. Arrays with more than two dimensions (e.g., `rubik = new Color[3][3][3]`) are not in the subset. "Ragged" arrays (e.g., `new int[3][]`) are not in the subset. In particular, students do not need to know that an `int[3][3]` really is an "array of arrays" whose rows can be replaced with other `int[]` arrays. However, students are expected to know that `arr[0].length` is the number of columns in a rectangular two-dimensional array `arr`. Anonymous arrays (e.g., `new int[] { 1, 2, 3 }`) are not in the subset.

15. The control structures `if, if/else, while, for` (including the enhanced `for` loop, also called the for-each loop), `return` are part of the AP Java subset. The `do/while, switch,` plain and labeled `break` and `continue` statements are not in the subset.

16. Methods: Method overloading (e.g., `MyClass.someMethod(String str)` and `MyClass.someMethod(int num))` is part of the AP Java subset. Students should understand that the signature of a method depends on the number, types, and order of its parameters but does not include the return type of the method.

    Methods with a variable number of parameters are not in the subset.

17. Classes: Students are expected to construct objects with the `new` operator, to supply construction parameters, and to invoke accessor and modifier methods. Students are expected to modify existing classes (by adding or modifying methods and instance variables). Students are expected to design their own classes.

18. Visibility: In the AP Java subset, all classes are `public.` All instance variables are `private.` Methods, constructors, and constants (`static final` variables) are either `public` or `private.` The AP Java subset does not use `protected` and package (default) visibility.

19. The AP Java subset uses `/* */, //,` and `/** */` comments. Javadoc comments `@param` and `@return` are part of the subset.

20. The `final` keyword is only used for `final` block scope constants and `static final` class scope constants. `final` parameters or instance variables, `final` methods, and `final` classes are not in the subset.

21. The concept of `static` methods is a part of the subset. Students are required to understand when the use of `static` methods is appropriate. In the exam, `static` methods are always invoked through a class, never an object (i.e., `ClassName.staticMethod(),` not `obj.staticMethod()).`

22. `static` variables are part of the subset.

23. The `null` reference is part of the AP Java subset.

24. The use of `this` is restricted to passing the implicit parameter in its entirety to another method (e.g., `obj.method(this)`) and to descriptions such as `"the implicit parameter this"`. Using `this.var` or `this.method(args)` is not in the subset. In particular, students are not required to know the idiom `"this.var = var",` where `var` is both the name of an instance variable and a parameter variable. Calling other constructors from a constructor with the `this(args)` notation is not in the subset.

25. The use of `super` to invoke a superclass constructor (`super(args)`) or to invoke a superclass method (i.e., `super.method(args)`) is part of the AP Java subset.

26. Students are expected to implement constructors that initialize all instance variables. Class constants are initialized with an initializer:

    ```
    public static final MAX_SCORE = 5;
    ```

    The rules for default initialization (with 0, `false` or `null`) are not in the subset. Initializing instance variables with an initializer is not in the subset. Initialization blocks are not in the subset.

27. Students are expected to extend classes and implement interfaces. Students are also expected to have a knowledge of inheritance that includes understanding the concepts of method overriding and polymorphism. Students are expected to implement their own subclasses.

28. Students are expected to read the definitions of interfaces and abstract classes and understand that the abstract methods need to be implemented in a subclass. Students are expected to write interfaces or class declarations when given a general description of the interface or class.

29. Students are expected to understand the difference between object equality (`equals`) and identity (==). The implementation of `equals` and `hashCode` methods are not in the subset.

30. Cloning is not in the subset, because of the complexities of implementing the `clone` method correctly and the fact that `clone` is rarely required in Java programs.

31. The `finalize` method is not in the subset.

32. Students are expected to understand that conversion from a subclass reference to a superclass reference is legal and does not require a cast. Class casts (generally from `Object` to another class) are part of the AP Java subset.

    The `instanceof` operator is not included in the subset, although it may be used in the context of the case study. Array type compatibility and casts between array types are not in the subset.

33. Students are expected to have a basic understanding of packages and a reading knowledge of `import` statements of the form

    ```
    import packageName.subpackageName.ClassName;
    ```

    `import` statements with a trailing `*`, static imports, packages and methods for locating class files (e.g., through a class path) are not in the subset.

34. Nested and inner classes are not in the subset.

35. The use of generic collection classes and interfaces are in the AP Java subset, but students need not implement generic classes or methods.

36. Enumerations, annotations, and threads are not in the subset.
37. Students are expected to understand the exceptions that occur when their programs contain errors (in particular, `NullPointerException`, `ArrayIndexOutOfBoundsException`, `ArithmeticException`, `ClassCastException`, `IllegalArgumentException`). Checked exceptions are not in the subset. In particular, the `try/catch/finally` statements and the `throws` modifier are not in the subset.

Summary Table

Tested in A Exam	Potentially relevant to CS1 course but not tested
`int`, `double`, `boolean`, `Integer.MAX_VALUE`, `Integer.MIN_VALUE`	`short`, `long`, `byte`, `char`, `float`
`+`, `-`, `*`, `/`, `%`, `++`, `--`, `=`, `+=`, `-=`, `*=`, `/=`, `%=`, `==`, `!=`, `<`, `<=`, `>`, `>=`, `&&`, `||`, `!`, and short-circuit evaluation	Using the values of `++`, `--` expressions in other expressions
`(int)`, `(double)`	`<<`, `>>`, `>>>`, `&`, `~`, `|`, `^`, `?:` Other numeric casts such as `(char)` or `(float)`
String concatenation	`StringBuffer`
Escape sequences `\"`, `\\`, `\n` inside strings	Other escape sequences (`\'`, `\t`, `\unnnn`)
`System.out.print`, `System.out.println`	`Scanner`, `System.in`, Stream input/output, GUI input/output, parsing input, formatted output

Tested in A Exam	Potentially relevant to CS1 course but not tested
	`public static void main(String[] args)`
1-dimensional arrays, 2-dimensional rectangular arrays	Arrays with 3 or more dimensions, ragged arrays
`if`, `if/else`, `while`, `for`, enhanced `for` (for-each), `return`	`do/while`, `switch`, `break`, `continue`
Modify existing classes, design classes	
`public` classes, `private` instance variables, `public` or `private` methods or constants	`protected` or package visibility
`@param`, `@return`	`javadoc`
`static` class variables	`final local` variables, `final` parameter variables, instance variables, methods or classes
`static` methods	`static` non-`final` variables
`null`, `this`, `super()`, `super(args)`, `super.method(args)`	`this.var`, `this.method(args)`, `this(args)`

Tested in A Exam	Potentially relevant to CS1 course but not tested
Constructors and initialization of `static` variables	Default initialization of instance variables, initialization blocks
Understand inheritance hierarchies. Design and implement subclasses. Modify subclass implementations and implementations of interfaces.	
Understand the concept of abstract classes and interfaces.	
Understand `equals`, `==`, and `!=` comparison of objects, `String.compareTo`	`clone`, implementation of `equals`, generic `Comparable<T>`
Conversion to supertypes and `(Subtype)` casts	`instanceof`
	Nested classes, inner classes enumerations
Package concept, `import packageName.className;`	`import packageName.*` `static import` defining packages, class path

Tested in A Exam	Potentially relevant to CS1 course but not tested
Exception concept, common exceptions	Checked exceptions `try/catch/ finally, throws`
`String, Math, Object, List, ArrayList`	Sorting methods in `Arrays` and `Collections`
Wrapper classes `(Integer, Double)`	autoboxing