

The AP Java Subset Changes April 2001

There were many comments submitted to the AP Computer Science Committee regarding the AP Java subset. The committee used several principles in determining which changes should be made to the subset.

1. The purpose of the AP Java subset is to specify what may appear on the **AP Computer Science Exam**. It is not a restriction on what teachers cover — it only restricts the features that can appear on the exam. Every instructor will need to include things in their course that will not be on the exam. The committee used the following guidelines to help determine what language features may appear on the AP Computer Science Exam.
 - a) The subset should be a “minimal subset.” Because of this, certain topics that are taught in different ways by different instructors are avoided, if possible.
 - b) If a certain syntactic concept is not important for writing questions on the exam, it is not included in the subset, even if it is very important to cover in a course that uses Java.
 - c) If there are several different ways to test a similar concept, the most straightforward way is preferred.
2. While there are many useful concepts that could be taught, everything can't be included in the A exam.

Suggested changes to the Java subset and the responses by the AP Computer Science Committee:

1. Add `char` to subset.

Response: After considerable discussion, the committee decided that while many (perhaps all) teachers would cover this topic, it was not essential for testing purposes, since for most questions that would use `char`, the same concepts could be tested with strings or integers. Only a few questions absolutely require that `char` be in the subset, and those questions would hinge on testing the distinction between `String` and `char` (e.g., `" " + 3` vs `' ' + 3`), and yield questions we would prefer to not ask in the first place.
2. Add the numeric cast `(double)` to subset.

Response: Added to subset. It will make questions dealing with real numbers easier.
3. Add I/O to subset.

Response: I/O is important in every course; however, many people do I/O in different ways. Because of this, I/O is not included in the subset. See Principle 1(a) above. There was a related suggestion of defining an IO class with static methods `readInt`, `readDouble`, `readString` that could be placed in the exam preamble (much like the list node classes). This suggestion was strongly considered but rejected on the grounds that this would have the same effect as defining an AP IO class.
4. Clarify statement on I/O, i.e., are console I/O and file I/O both not part of subset?

Response: The only I/O that is part of the subset is `System.out.print` and `System.out.println`.
5. Add static variables to the subset.

Response: Static final variables have been added to the subset. The case study author may elect to incorporate static nonfinal variables into the subset to solve a typical problem that these variables are used for (namely the use of a variable to count the number of active instances of a class). If so, static nonfinal variables may be testable in the context of the case study only.
6. Why do you have restrictions on the use of `super`?

Response: `super` as part of the constructor was in the subset. In response to this suggestion, the use of `super.method()` has been added to the subset, but in the AB course only. See

- principle 2 above. Using `super` to access inherited data members is not in the subset, because all data members are private.
7. Add `this` to subset.
Response: `this` is pedagogically useful for introducing concepts but is not tested, see Principle 1 above.
 8. Add inner classes to subset.
Response: Inner classes are very useful, especially to instructors who elect to cover GUI programming. However, they are not essential for testing purposes, and are not in the subset.
 9. Remove various Java APIs, such as `ArrayList`, `List`, etc.
Response: The committee feels strongly that data structures are part of the AB curriculum. We feel that it is better to use standard Java Collections classes than to write our own.
 10. Remove javadoc from subset as javadoc is not part of java.
Response: javadoc comments have been removed from the subset. There was concern expressed because `@precondition` was not a part of javadoc.
 11. Change `Stack`, `Queue`, and `PriorityQueue` interfaces to match the Collections API.
Response: These interfaces exist solely for the purposes of writing questions that manipulate stacks, queues, and priority queues, and as such, we find it preferable to use the standard names. It is unlikely that students will have any problem with `push`, `pop`, `enqueue`, `dequeue`, etc. The Committee did, however, remove `size`, and add `isEmpty`.
 12. Change implementation of `ListQueue`.
Response: First, observe that the provided implementation of `ListQueue` is not part of the subset and is not directly testable. Nonetheless, other changes to `LinkedList` allow for a simplification of the `ListQueue` implementation, by using `removeFirst`, `getFirst`, and `addLast`. It was felt that this would be a natural exercise, and thus provided the impetus for the addition to the `LinkedList` class. The subset also specifies that the students will be provided with a quick reference sheet described the classes, interfaces, and methods from the Java API that are required. Included in the reference will also be information about the inheritance hierarchy. This means that students will be required to know that `addFirst` is part of `LinkedList`, but not part of `List` or `ArrayList`, and will provide the committee the opportunity to test various inheritance facts without introducing a set of artificial classes.
 13. Use `Math.Random` instead of `java.util.Random`.
Response: `java.util.Random` gives random numbers as both integers and reals in the unit range, and is therefore preferable.
 14. Threads should be in subset.
Response: not in most introductory courses
 15. Add GUI's to subset.
Response: Shouldn't be in subset. There are many different ways of doing GUI's see principle 1(a) above.
 16. Take AP classes out of subset.
Response: There are no AP classes in the subset. All standard classes are part of the Java API. The subset does describe the notion of a "standard nonstandard class," such as `ListNode` and `TreeNode`. The idea of this is that in the current exam, every time a `TreeNode` is declared (data, left and right children), we must put a declaration on the question page. If there are three such questions, the same code appears three times. It is important to realize that students, who are operating under time pressure, have to read the entire question. As a result, it would be preferable to state at the start of the exam what a `TreeNode` is, so that each question can simply refer back to it. That is exactly what is being done with `ListNode`, `TreeNode`, and also the `Stack`, `Queue`, and `PriorityQueue` interfaces. The standard Java API classes required for the exam will be summarized at the end of the exam in a quick reference sheet. The standard

- nonstandard classes, such as `TreeNode`, will be listed at the front of the exam, along with other disclaimers that simplify the presentation of the exam, and the entire set of disclaimers will be published in advance of the exam. An example of another simplifying mechanism is that if we define a single static method that computes something simple, it will not be placed in a surrounding class declaration.
17. Fix `ListNode` and `TreeNode` classes.
Response: There was concern that these classes were overly object-oriented because the data was private, and that in a real application, the data would be public, and perhaps the classes themselves would be nested. Using nested classes was not deemed useful for testing purposes, and in fact would likely complicate the presentation. Furthermore, there was significant agreement that although designers may well decide that there are legitimate reasons to use public data, those are decisions that should not be presented to students in the first two courses. Additionally, doing so would undermine our ability to ask design questions. For example, the committee would not be able to ask a question whose answer relied on the fact that good design requires private data.
 18. Omit combined arithmetic assignment operators.
Response: Leave in the subset. They can shorten the amount of code in a problem, which can often be helpful on the exam.
 19. There is too much emphasis on Data Structures in the exam.
Response: Remember that there are two AP Computer Science Exams, A and AB, which correspond to CS1 and CS2 respectively. The AB exam has an emphasis on data structures; however, you don't want to compare this exam with your CS1 course.
 20. High school teachers need more resources for the switch to Java.
Response: The College Board is very concerned about the need for training for high school teachers and is currently working on several initiatives to find creative ways to increase training opportunities for high school teachers. As opportunities become available, they will be posted on the College Board Web site or the discussion group (online mailing list).
 21. The AP courses should better match industry and the real world.
Response: The purpose of the AP courses is to provide high school students with an opportunity to get college credit or placement. Because of this purpose, AP strives to match what most colleges are teaching and would only match industry to the extent that the majority of colleges CS1/CS2 match what is being done in industry.
 22. Some people commented on the language choice of the exam. Most of these comments from college faculty were favorable to the choice of Java as the language; however, there were a few people who were critical of this change.
Response: The surveys that have been done show that many colleges are moving to Java. More details on this can be found in the rationale for the switch.

Additional subset changes made by the AP Computer Science Committee

Item #16 was changed from the original in two ways. First, the committee clarified that students should understand the meaning of the signature of a method, and the fact that the return type is not part of the signature. This was specifically added because knowing this is important for overriding methods and implementing interfaces. Even though the rule is the same in C++, the AP C++ Subset does not require students to understand the role of the return type. Second, the phrasing about trick questions was removed because trick questions are never asked. Leaving the disclaimer in may give some the impression that although trick questions won't be asked about overload resolution, there might be other places where trick questions would be asked.

The Standard Classes, Interfaces, and Methods section was overhauled to make clear the properties of the classes that are testable (for instance, immutability of `String`, `Integer`, `Double`), as well as the inheritance hierarchies. Arrays and Collections were removed, since they add little. In `LinkedList`, six

methods were added that are useful for implementing double-ended queues (`add`, `remove`, `get` at both front and back.)

The committee reworded many parts of the subset. In particular, the preamble was reworded in an attempt to make clear that what is being described is simply a “testable subset,” and to make clear that all courses are expected to cover more than the minimal subset.