

*Third Edition
with GridWorld*

Be Prepared
for the

AP

**Computer
Science
Exam in Java**

Maria Litvin

Phillips Academy, Andover, Massachusetts

Practice exam contributors:

Roger Frank

Ponderosa High School, Parker, Colorado

Judith Hromcik

Arlington High School, Arlington, Texas

Gary Litvin

Skylight Publishing, Andover, Massachusetts

Dave Wittry

Taipei American School, Taiwan

Skylight Publishing
Andover, Massachusetts

**Copyright © 2007 by
Maria Litvin and Skylight Publishing**

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author and Skylight Publishing.

Library of Congress Control Number: 2007901348

ISBN 978-0-9727055-6-1

Skylight Publishing
9 Bartlet Street, Suite 70
Andover, MA 01810

web: <http://www.skylit.com>
e-mail: sales@skylit.com
support@skylit.com

1 2 3 4 5 6 7 8 9 10 12 11 10 09 08 07

Printed in the United States of America

Chapter 1. Exam Format, Grading, and Tips

1.1. Exam Format and Materials

Figure 1-1 shows the format of the AP Computer Science exam. The exam takes 3 hours of test time (plus breaks and time for instructions). It is divided into two sections. Section I consists of 40 multiple-choice questions with a total allotted time of 1 hour and 15 minutes (1.5 to 2 minutes per question on average). Section II consists of four free-response questions with a total allotted time of 1 hour and 45 minutes (20-30 minutes per question). The free-response questions usually consist of two or three parts each. No computers, calculators, other devices, books, or materials are allowed.

Copyright © 2007 by Skylight Publishing

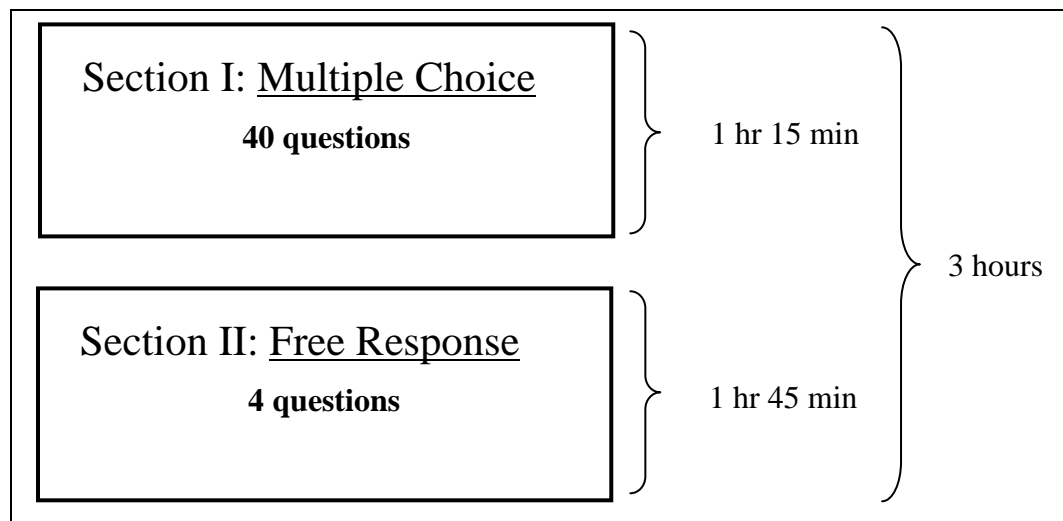


Figure 1-1. AP Computer Science exam format

Exam materials given to you at the exam will include a booklet containing the needed case study code and *Quick Reference* — a list of the library classes and their methods included in the AP subset [as well as the `ListNode` and `TreeNode` classes]. These materials are provided for reference — it is expected that you will already be very familiar and comfortable with the case study and the required library classes before the exam. <http://www.skylit.com/beprepared/> has current links to these materials.

The multiple-choice section is a mixture of questions related to general computer science terms, program design decisions, specific elements of Java syntax, properties of classes, logical analysis of fragments of Java code, OOP concepts, and five-six questions related to the case study. 「 The AB exam also includes questions about data structures: linked lists, stacks, queues, trees, sets, maps, priority queues, hash tables, heaps, etc., and “Big-Oh” analysis of algorithms. 」

The free-response questions usually aim to cover a wide range of material: arrays, strings, classes and interfaces, Java library classes (within the AP subset), and so on. 「 The AB exam may include questions on two-dimensional arrays, linked lists, trees, sets, maps, and other data structures. 」 In past exams, students have not been asked to write complete programs. Usually, they were asked to write a constructor or a method that performs a specified task under a given header for the method. The second part of the question often refers to the class or method implemented in the first part, but each part is graded separately, and your implementation of Part (a) does not have to be correct in order for you to get full credit for Part (b). Part (c) may ask questions about your implementation or ask you to write an additional method that uses Parts (a) and/or (b). In that case you are to assume that the methods in the previous parts work as intended, regardless of what you wrote for them.

Free-response questions usually also include a “design” question, in which you are asked to design a small class, then write it or some fragments of it or just use it in other parts of the question. Your design will be graded based on the appropriateness of the features of your class, appropriate names for methods and variables, and other criteria.

One free-response question is based on the *Case Study*. It may ask you to extend a *Case Study* class and to write a new method or rewrite an existing method.

1.2. The Java Subset

The Development Committee has defined a restricted subset of Java to be tested on the exams. The purpose of the subset is to focus the AP CS program more on general concepts than on the specifics of Java and to limit the scope, especially of material related to the peculiarities of Java. The subset is described in The College Board’s *Advanced Placement Course Description for Computer Science*; we have a link to it from this book’s web site <http://www.skylit.com/beprepared/>. 「 The AB “superset” also defines the `ListNode` and `TreeNode` classes (see Chapter 6). 」

What is in the subset? Actually, quite a bit.

A exam:

- boolean, int, and double primitive data types. (int) and (double) casts. **Other primitive data types, including char, are not in the subset and should be avoided on exams.**
- Assignment (=), arithmetic (+, -, *, /, %), increment (++), decrement (--), compound assignment (+=, -=, *=, /=, %=), relational (<, >, <=, >=, ==, !=), and logical (&&, ||, !) operators. **Use only the postfix form of ++ and -- (x++ or x--), and do not use them in expressions.**
- + and += operators for concatenating strings. String's compareTo, equals, length, substring, and indexOf(String s) methods. \n, \\, and \" escape sequences in literal strings.
- System.out.print and System.out.println.
- One-dimensional arrays, array.length, arrays of objects, initialized arrays such as `int[] x = {1,2,3};`
- if-else, for (including the “for each” form, `for(type x : values)...`), while, return. **But do-while and switch are not included.**
- Classes. Constructors, the new operator, public and private methods, static methods, static final variables (constants), overloaded methods, null. **All instance variables are private.** Default initialization rules are not in the subset and won't come up on the exam.
- Inheritance, interfaces and abstract classes, extends, implements. Calling a superclass's constructor from a subclass (as in `super(...)`). Calling a superclass's method from a subclass (as in `super.someMethod(...)`). Passing this object to a method (as in `otherObject.someMethod(this)`).
- NullPointerException, ArrayIndexOutOfBoundsException, ArithmeticException, IllegalArgumentException, ClassCastException.
- Library classes and methods:
 - String:** length(), substring(...), indexOf(String s)
 - Integer:** Integer(int x), intValue()
 - Double:** Double(double x), doubleValue()
 - Math:** abs(double x), pow(double base, double exp), sqrt(double x), random()
 - Random:** nextInt(int n), nextDouble()

Also understand toString methods for all objects, the Comparable<T> interface and compareTo, equals and compareTo for String, Integer, and Double.

- ArrayList<E> (see Section 2.6).

┌ AB exam:

All of the above, plus you should be able to:

- Write code that throws exceptions, such as


```
throw new NoSuchElementException();
throw new IllegalArgumentException();
```
- Design and implement interfaces, abstract classes, subclasses.
- Use small subsets of methods for collection interfaces and classes `List<E>`, `ArrayList<E>`, `LinkedList<E>`, `Iterator<E>`, `ListIterator<E>`, `Stack<E>`, `Queue<E>`, `PriorityQueue<E>`, `Set<E>`, `TreeSet<E>`, `HashSet<E>`, `Map<K,V>`, `TreeMap<K,V>`, `HashMap<K,V>` (see Chapter 6).

└

If you feel you must stray from the subset in your free-response solution, you might have misunderstood the problem and be making it harder than it is.

Things that are not in the AP subset and should be avoided include the following:

- Java syntax abominations, such as the `?_:_` operator and the “comma” operator
- `++` and `--` in expressions (as in `a[i++]`)
- Primitive data types other than `boolean`, `int`, and `double` (`char` is not in the subset)
- All bit-wise logical operators

Also not in the subset and will not be tested:

- The `switch` statement, the `do-while` loop, `continue` in loops
- The prefix form of `++` and `--` operators (`++k`, `--k`)
- Library classes (such as `StringBuffer`, `Arrays`, `DecimalFormat`, etc.), unless specifically listed in the subset
- checked exceptions and `try-catch-finally` statements
- `System.in` and `Scanner`; any input and output other than `System.out.print` and `System.out.println`
- `enum` data types

1.3. Grading

The exams are graded on a scale from 1 to 5. Grades of 5 and 4 are called “extremely well qualified” and “well qualified,” respectively, and usually will be honored by colleges that give credit or placement for AP exams in computer science. A grade of 3, “qualified,” especially on the A exam, may be denied credit or placement at some colleges. Grades of 2, “possibly qualified,” and 1, “no recommendation,” will not get you college credit or placement.

Table 1-1 presents published statistics and grade distributions on the 2004 A and AB exams. In 2004, 13,834 candidates took the A exam and 5,807 candidates took the AB exam.

	Computer Science A		Computer Science AB	
	Number	%	Number	%
Students	13,834	100.0	5,807	100.0
Grade:				
5	2,572	18.6	1,574	27.1
4	3,270	23.6	1,058	18.2
3	2,102	15.2	1,020	17.6
2	1,309	9.5	702	12.1
1	4,581	33.1	1,453	25.0
3 or Higher	7,944	57.4	3,652	62.9

Table 1-1. 2004 grade distributions for A and AB exams

The multiple-choice and free-response sections weigh equally in the final grade.

The College Board uses a weighted combination of the multiple-choice (MC) and free-response (FR) scores to determine the final total score:

$$\text{totalScore} = \text{MC_coeff} * (\text{countCorrect} - 0.25 * \text{countWrong}) + \text{FR_coeff} * \text{FR_score};$$

For multiple-choice questions, one point is given for each correct answer and 1/4 point is subtracted for each wrong answer. Free-response questions are graded by a group of high school teachers and college professors. Scores are based on a *rubric* established by the Chief Reader, Exam Leader, and Question Leaders. Each free-response question is graded out of 9 points, with partial credit given according to the rubric.

The final score is obtained by adding the MC and FR weighted scores. The MC and FR coefficients are chosen in such a way that they give equal weights to the multiple-choice and free-response sections of the exam. For example, if the exam has 40 multiple-choice questions and 4 free-response questions, weights of 1.25 for multiple-choice and 1.3889 for free-response will give each section a maximum total of 50, for a maximum possible total score of 100.

Four cut-off points determine the grade. Table 1-2 shows the maximum composite scores and cut-off points used for the 1999 and 2004 exams. In 2004, 79% or more correct answers on the A exam and 68% or more correct answers on the AB exam would get you a 5. The cut-off points are determined by the Chief Reader and may vary slightly from year to year based on the score distributions and close examination of a sample of individual exams.

		A Max composite score 80 (1.00 * MC + 1.1111 * FR)		AB Max composite score 100 (1.25 * MC + 1.3889 * FR)	
AP Grade		1999	2004	AP Grade	
5		60 - 80	63 - 80	5	70 - 100
4		45 - 59	49 - 62	4	60 - 69
3		33 - 44	39 - 48	3	41 - 59
2		25 - 32	32 - 38	2	31 - 40
1		0 - 24	0 - 31	1	0 - 30

Table 1-2. Score-to-grade conversion

Copyright © 2007 by Skylight Publishing

1.4. A or AB?

Table 1-1 shows that a larger percentage of AB exam takers got a 5. That's how it should be — if you don't feel that you can get a 4 or 5 on the AB exam, you don't need to take it. If you haven't covered all the AB material or are not comfortable with it, take the A exam. It makes little sense to get a 2 or 1 on the AB exam if you could get a 5 or 4 on the A exam. The practice exams in this book will help you make up your mind.

Statistical analysis of published results from the 2004 exam shows that over 98% of students who got at least 27 out of 40 on the multiple-choice section received a 4 or a 5 for the whole exam. This may or may not be true for our practice exams. You will know only after the exam!

Most colleges will take your AP courses and exam grades into account in admissions decisions if you take your exams early enough. But acceptance of AP exam results for credit and/or placement varies widely among colleges. In general, the A exam corresponds to a CS-1 course (Introductory Computer Science or Computer Programming I), a one-semester course for computer science majors. The AB exam corresponds to CS-1 + CS-2; that is, the first programming course plus a course on data structures, usually a one-year sequence for computer science majors. Some colleges give one-semester credit for the A exam and two-semester credit for the AB exam, as intended. But other colleges may only give one semester credit, regardless of the exam. They may also base their decision on your grade. For example, you may get a full year credit only if you got a 5 on the AB exam. Some colleges may not give any credit at all.

The AP program in computer science is a rigorous and demanding program that is comparable to or exceeds the level of the respective first-year computer science courses at most colleges.

If you plan to major in computer science and your college of choice does not recognize a good grade on the AP exam for credit and/or placement, you should examine the reasons carefully. Decide for yourself whether these reasons are valid or just stem from the bias of that college or its computer science department.

But if the college that you definitely want to attend does not give any admissions preference or additional credit for the AB exam, it may be better to focus on getting a 5 on the A exam.

To do well on the AB exam, you have to know linked lists, binary trees, stacks and queues, priority queues, hashing, sets and maps, and Big-Oh analysis of algorithms.

If you know this material, you shouldn't be afraid of the AB exam. Don't assume that it is "just harder." The AB exam simply includes more topics. It does not ask you to write code better or faster.

The A and AB exams share many multiple-choice questions, and, once you learn the data structures part, the AB exam questions are not necessarily harder than the questions on the A exam. The AB exam questions have to be more diverse in order to cover all the material in the same number of questions; this may actually make the exam easier for you if you have studied all the AB material. For example, you may enjoy recursive handling of binary trees, but be prone to mistakes in programs that involve iterations and arrays.

1.5. Exam Taking Tips

Some things are obvious:

- If you took the time to read a multiple-choice question and all the answer choices but decided to skip it, take an extra ten seconds and guess. Most likely you have eliminated one or two wrong answers even without noticing.
- If a common paragraph refers to a group of questions and you took the time to read it, try each question in the group.
- Do read the question before jumping to the code included in the question.

There are a few important things to know about answering free-response questions.

Remember that all free-response questions have equal weight. Don't assume that the first question is the easiest and the last is the hardest.

In a nutshell: be neat, straightforward, and professional; keep your exam reader in mind; don't show off.

More specifically:

Stay within the AP Java subset.

Remember that the elegance of your code does not count. More often than not, a brute-force approach is the best. You may waste a lot of time writing tricky, non-standard code and trick yourself in the process or mislead your exam reader who, after all, is only human. No one will test your code on a computer.

Superior efficiency of your code does not count, unless the desired Big-Oh of the solution is specifically stated in the question.

Remember that Parts (b) and (c) of a question are graded independently from the previous parts, and may actually be easier: Part (a) may ask you to write a method, while Part (b) or Part (c) may simply ask you to use it.

It is common for method(s) specified in Part (a) to be called in subsequent parts. Do so, even if your Part (a) is incorrect or left blank. Do not re-implement code from earlier parts in later parts — you will waste valuable time and may lose points for doing so.

5. Bits of “good thinking” count. You may not know the whole solution, but if you have read and understood the question, go ahead and write fragments of code that may earn you partial credit points. But don’t spend too much time improvising incorrect code.
6. Don’t waste your time erasing large portions of work. Instead, cross out your work with one neat line, but only after you have something better to replace it with. Do not cross out a solution if you have no time to redo it, even if you think it is wrong. You won’t be penalized for incorrect code and may get partial credit for it. Exam readers are instructed not to read any code that you have crossed out. But if you wrote two solutions, be sure to cross one out: otherwise only the first one on the page will be graded.
7. Read the comment above the method header quickly — it usually restates the task in a more formal way and sometimes gives hints. Assume that all preconditions are satisfied — don’t add unnecessary checks to your code.
8. One common mistake is to forget a `return` statement in a non-void method. Make sure the returned value matches the specified type.
9. Do not ignore any hints in the question description. If an algorithm is suggested for a method (as in “you may use the following algorithm”), don’t fight it, just do it! If the description says “you may use a helper method,” be sure to write and use one: chances are it is much more difficult to come up with a solution without a helper method.
10. Remember that the exam readers grade a vast number of exams in quick succession during a marathon grading session every June. Write as neatly as possible. Space out your code (don’t save paper).
11. Always indent your code properly. This helps you and your exam reader. If you miss a brace but your code is properly indented, the reader (as opposed to a Java compiler) may accept it as correct. Similarly, if you put each statement on a separate line, a forgotten semicolon may not be held against you.
12. Follow the Java naming style: the names of all methods, variables, and parameters start with a lowercase letter. Use meaningful, but not too verbose, names for variables. `count` may be better than `a`; `sum` may be better than `temp`; `row`, `col` may be better than `i`, `j`. But `k` is better than `loopControlVariable`. If the question contains examples of code with names, use the same names when appropriate.

13. Don't bother with comments; they do not count and you will lose valuable time. Occasionally you can put a very brief comment that indicates your intentions for the fragment of code that follows. For example:

```
// Find the first empty seat:  
...  
...
```

14. Don't worry about `imports` — assume that all the necessary library classes are imported.
15. Code strictly according to the specifications and preconditions. Avoid extraneous “bells and whistles” — you will lose points. Never add `System.out.print/println` in solutions unless specifically asked to do so.
16. Use recursion when appropriate: [often in methods that deal with binary trees; otherwise] if specifically requested or especially tempting.
17. Don't try to catch the exam authors on ambiguities: there will be no one to hear your case, and you'll waste your time. Instead, try to grasp quickly what was meant and write your answer.
18. Don't quit until the time is up. Use all the time you have and keep trying. The test will be over before you know it.