**Computer Science 12 AP**

41 Things You Should Know

Here's a list of 41 things you should know for the AP exam and this courses final exam. If it is listed here, there is a 99% chance that it will be on the final exam.

1. when to use integers, doubles, and booleans as variables
2. how math operations like casting, dividing, and modulus work with integers and doubles
3. number limitations, number imprecision and roundoff errors
4. conversion from decimal to binary to hexadecimal
5. conditional statements with if, else if, and else, && and ||
6. compare boolean conditions and offer alternative boolean conditions
7. iterate using for, while, do while, and for each
8. read recursive methods
9. limitations when using the for each loop
10. force user to continue entering values until appropriate criteria are met
11. read and write methods with appropriate return types and parameters, explain what overloading a method means
12. find random values using Math.random() eg. int x = (int)(Math.random() \*100) ,   
    use Math class static methods like exp, pow
13. create an array of primitive types or objects ( eg. Student[] studs = new Student[1000] )
14. common array algorithms like sequential search, binary search, selection sort, insertion sort, merge, mergesort, maximum and minimum value, average, sum, and how these algorithms may be applied to any list type data structure (eg. ArrayList)
15. common array algorithms that require resizing (adding, inserting, deleting)
16. creating an ArrayList using ArrayList<E> name = new ArrayList<E>(), how to use the ArrayList class, compare/contrast ArrayLists and arrays, apply ArrayLists in your own programs and in modification of the case study
17. how to use common String class methods, common String algorithms like joining Strings in a pattern, grabbing portions of Strings based on a pattern, checking a String for simple patterns, counting occurences of a String within a String
18. what is meant by Strings being immutable, how to check equality of String text versus equality of String memory address
19. how to convert between integers, doubles, and Strings
20. how to use the Integer and Double class to store primitive values for object based lists
21. how parameters such as primitives, objects, and Strings are changed/not changed permanently by methods
22. informal comparisions of the time it takes various algorithms to complete
23. the difference between a runtime and compile error and examples of each
24. what an exception is, what an assertion is, what a pre and post condition is
25. how to declare constant variables
26. the difference between top down design versus bottom up design
27. how static variables are different than regular class variables, what the requirements are for a static method and how static methods may be called without creating an instance
28. the difference between a class, an abstract class, an interface, and a runner class
29. how to implement an interface, why interfaces are useful in making code more versatile, examples of interfaces in the case study, examples of interfaces with List interface, why you can't instantiate an interface, how a class can implement several interfaces
30. how to code the comparable interface for a class, the reason Object is used as a parameter and how to cast the Object parameter properly, why an interface like comparable is useful
31. why you might use an abstract class
32. how to extend a class to add or modify behavior
33. the affect of public and private member declaration, use of accessor and mutator methods
34. what is overriding a method and give examples of how this might be done in the case study
35. recognize and use class constructor/s when creating instances
36. proper use of super() in subclasses
37. polymorphism rules for creating instances. eg ClassA temp = new ClassB()  
    polymorphism rules for casting. eg ((ClassB)temp).classbmethod()  
    polymorphism rules for parameters. eg BoxBug as method parameter in place of a Bug  
    polymorphism rules for overloaded methods. eg overloaded methods  
    how polymorphism can give your code more versatility
38. how to modify the case study using the boundedgrid, actor, and location classes  
    (This is a ¼ of the written! and 10% of the multiple choice)
39. detailed knowledge of how the Bug, Critter, ChameleonCritter, and CrabCritter are coded
40. terms like encapsulation, information hiding, and procedural abstration and how they relate to classes
41. toString method for system outing class information
42. each method of every class listed in the AP subset reference guide so that you can code, compile, and smile without wasting alot of time reading the reference guide

What will not be on the term test or the AP exam that we have seen this year:  
  
File Reading/Writing. Netbeans GUI. UserInput class.

Random class. RFID Phidgets. Robocode. Lego NXT.