Chapter 4: Creating and Using Arrays

1. B. Three dots (. . .) are the syntax for a method parameter of type varargs. It is treated like an array.
2. B. Array indexes are zero based in Java. A varargs parameter is simply another way of passing in data to a method. From within the method, it is treated just like you had written `Frisbee[] f` as the method parameter. Therefore, the first element uses the 0th index, and Option B is correct.
3. D. Trick question! While `int` is a primitive, all arrays are objects. One way to tell is that an array has a public instance variable called `length`. Another way is that you can assign it a variable of type `Object`. Therefore, Option D is correct.
4. C. The array braces are allowed to appear before or after the variable name, making the `tiger` and `bear` declarations correct. The braces are not allowed to appear before the type making the `lion` declaration incorrect. Therefore, Option C is correct.
5. C. From within a method, an array or varargs parameter is treated the same. However, there is a difference from the caller's point of view. A varargs parameter can receive either an array or individual values, making Options A and B compile. However, an array parameter can only take an array, which prevents Option C from compiling.
6. A. Arrays use the `length` variable to determine the number of elements, making Option A correct. For an `ArrayList`, Option D would have been the answer.
7. C. A two-dimensional array is declared by listing both sizes in separate pairs of braces. Option C correctly shows this syntax.
8. B. There is nothing wrong with this code. It correctly creates a seven-element array. The loop starts with index 0 and ends with index 6. Each line is correctly output. Therefore, Option B is correct.
9. B. Sorry. This is just something you have to memorize. The `sort()` and `binarySearch()` methods do sorting and searching, respectively.
10. B. The elements of the array are of type `String` rather than `int`. Therefore, we use alphabetical order when sorting. The character 1 sorts before the character 9, alphabetically making Option A incorrect. Shorter strings sort before longer strings when all the other characters are the same, making Option B the answer.
11. B. Array indices start with 0, making Options C and D incorrect. The `length` attribute refers to the number of elements in an array. It is one past the last valid array index. Therefore, Option B is correct.
12. C. When using an array initializer, you are not allowed to specify the size separately. The size is inferred from the number of elements listed. Therefore, `tiger` and `ohMy` are incorrect. When you're not using an array initializer, the size is required. An empty array initializer is allowed. Option C is correct because `lion` and `bear` are legal.

13. B. Since no elements are being provided when creating the arrays, a size is required. Therefore, `lion` and `bear` are incorrect. The braces containing the size are required to be after the type, making `ohMy` incorrect. The only one that is correct is `tiger`, making the correct answer Option B.

14. C. The `binarySearch()` method requires a sorted array in order to return a correct result. If the array is not sorted, the results of a binary search are undefined.

15. A. An `ArrayList` expands automatically when it is full. An array does not, making Option A the answer. The other three statements are true of both an array and an `ArrayList`.

16. C. This code creates a two-dimensional array of size 1×2. Lines `m1` and `m2` assign values to both elements in the outer array. Line `m3` attempts to reference the second element of the outer array. Since there is no such position, it throws an exception, and Option C is correct.

17. B. The code sorts before calling `binarySearch()`, so it meets the precondition for that method. The target string of `"Mac"` is the second element in the sorted array. Since array indices begin with zero, the second position is index 1, and Option B is correct.

18. A. A multi-dimensional array is created with multiple sets of size parameters. The first line should be `char[] ticTacToe = new char[3][3];`. Therefore, Option A is the answer.

19. B. The first line creates one object; the array itself. While there are four references to `null` in that array, none of those are objects. The second line creates one object and points one of the array references to it. So far there are two objects: the array itself and one object it is referencing. The third line does the same, bringing up the object count to three. Therefore, Option B is correct.

20. B. As with a one-dimensional array, the braces must be after the type, making `alpha` and `beta` illegal declarations. For a multi-dimensional array, the braces are allowed to be before and/or after the variable name. They do not need to be in the same place. Therefore, the remaining three are correct, and Option B is correct.

21. B. Options A, C and D represent 3x3 2D arrays. Option B best represents the array in the code. It shows there are three different arrays of different lengths.

22. D. `names.length` is the number of elements in the array. The last valid index in the array is one less than `names.length`. In Java, arrays do not resize automatically. Therefore, the code throws an `ArrayIndexOutOfBoundsException`.

23. C. The code `days.size()` would be correct if this was an `ArrayList`. Since it is an array, `days.length` is the correct code. Therefore, the code does not compile, and Option C is the answer.

24. C. Since the braces in the declaration are before the variable names, the variable type `boolean[][][]` applies to both variables. Therefore, both `bools` and `moreBools` can

reference a 3D array.

25. C. Calling `toString()` on an array doesn't output the contents of the array, making Option C correct. If you wanted Option A to be the answer, you'd have to call `Arrays.toString(strings)`.

26. B. Arrays begin with an index of 0. This array is a 3×3 array. Therefore. only indexes 0, 1. and 2 are valid. Line `r2` throws an `ArrayIndexOutOfBoundsException`. Therefore. Option B is correct.

27. D. Three dots in a row is a varargs parameter. While varargs is used like an array from within the method, it can only be used as a method parameter. This syntax is not allowed for a variable, making Option D the answer.

28. D. Line 6 assigns an `int` to a cell in a 2D array. This is fine. Line 7 casts to a general `Object[]`. This is dangerous, but legal. Why is it dangerous, you ask? That brings us to line 8. The compiler can't protect us from assigning a `String` to the `int[]` because the reference is more generic. Therefore, line 8 throws an `ArrayStoreException` because the type is incorrect, and Option D is correct. You couldn't have assigned an `int` on line 8 either because `obj[3]` is really an `int[]` behind the scenes and not an `int`.

29. C. The code sorts before calling `binarySearch`, so it meets the precondition for that method. The target string of `"RedHat"` is not found in the sorted array. If it was found, it would be between the second and third element. The rule is to take the negative index of where it would be inserted and subtract 1. It would need to be inserted as the third element. Since indexes are zero based, this is index 2. We take the negative, which is -2, and subtract 1, giving `-3`. Therefore, Option C is correct.

30. B. Array indexes begin with zero. `FirstName` is the name of the class, not an argument. Therefore, the first argument is `Wolfie`, and Option B is correct.

31. C. The name of the program is `Count` and there are two arguments. Therefore, the program outputs `2`, and Option C is correct.

32. B. This class is called with two arguments. The first one (`seed`) is stored in the variable `one`. Then the array is sorted, meeting the precondition for binary search. Binary search returns `1` because `seed` is the second element in the sorted array, and Java uses zero-based indexes. Option B is correct.

33. D. Options A and B show the braces can be before or after the variable name and produce the same array. Option C specifies the same array the long way with two arrays of length 1. Option D is the answer because it is different than the others. It instead specifies an array of length 1 where that element is of length 2.

34. C. Arrays are indexed using numbers, not strings, making Options A and B incorrect. Since array indexes are zero based, Option C is the answer.

35. D. In Java, arrays are indexed starting with 0. While it is unusual for the loop to start with 1, this does not cause an error. What does cause an error is the loop ending at

`data.length`, because the `<=` operator is used instead of the `<` operator. The last loop index is 6, not 7. On the last iteration of the loop, the code throws an `ArrayIndexOutOfBoundsException`. Therefore, Option D is correct.

36. C. Array indexes begin with zero. FirstName is the name of the class, not an argument. The first and only argument is Wolfie. There is not a second argument, so Option C is correct.

37. D. This code is correct. Line `r1` correctly creates a 2D array. The next three lines correctly assign a value to an array element. Line `r3` correctly outputs `3 in a row!`

38. D. Arrays expose a `length` variable. They do not have a `length()` method. Therefore, the code does not compile, and Option D is correct.

39. B. This one is tricky since the array braces are split up. This means that `bools` is a 3D array reference. The braces both before and after the variable name count. For `moreBools`, it is only a 2D array reference because there are only two pairs of braces next to the type. In other words, `boolean[][]` applies to both variables. Then `bools` gets another dimension from the braces right after the variable name. However, `moreBools` stays at 2D, making Option B correct.

40. B. Since no arguments are passed from the command line, this creates an empty array. Sorting an empty array is valid and results in an empty array. Therefore, Option B is correct.

41. D. Java requires having a sorted array before calling `binarySearch`. Since the array is not sorted, the result is undefined, and Option D is correct. It may happen that you get `1` as the result, but this behavior is not guaranteed. You need to know for the exam that this is undefined even if you happen to get the "right" answer.

42. B. Line 8 attempts to store a `String` in an array meant for an `int`. Line 8 does not compile, and Option B is correct.

43. A. This array has two elements, making `listing.length` output `2`. While each array element does not have the same size, this does not matter because we are only looking at the first element. The first element has one. This makes the answer Option A.

44. C. `FirstName` is the name of the class, not an argument. There are no other arguments, so `names` is an empty array. Therefore, Option C is correct.

45. A. In Java, arrays are indexed starting with 0. While it is unusual for the loop to start with 1, this does not cause an error. It does cause the code to output six lines instead of seven since the loop doesn't cover the first array element. Therefore, Option A is correct.

46. B. The name of the program is `Count`, and there is only one argument because double quotes are used around the value. That argument is a `String` with three characters: `1`, a space, and `2`. Therefore, the program outputs `1`, and Option B is correct.

47. A. Java requires having a sorted array before calling `binarySearch()`. You do not have

to call `Arrays.sort` to perform the sort though. This array happens to already be sorted, so it meets the precondition. The target string of `"Linux"` is the first element in the array. Since Java uses zero-based indexing, the answer is Option A.

48. A. From within a method, an array parameter and a varargs parameter are treated the same. From the caller, an array parameter is more restrictive. Both types can receive an array. However, only a varargs parameter is allowed to automatically turn individual parameters into an array. Therefore, statement I is correct and the answer is Option A.

49. B. All of the variables except `nums2b` point to a 4D array. Don't create a 4D array; it's confusing. The options show the braces can be before or after the variable in any combination. Option B is the answer because `nums2b` points to a 3D array. It only has three pairs of braces before the variable and none after. By comparison, `nums2a` has three pairs of braces before the variable and the fourth pair of braces after.

50. C. Binary search returns an `int` representing the index of a match or where a match would be. An `int` cannot be stored in a `String` variable. Therefore, the code does not compile and the answer is Option C.