

Score: 0 / 30 (0%)

AP Chapter 6 Test

MULTIPLE CHOICE

- ☒ 1. For questions 1-2, assume values is an int array that is currently filled to capacity, with the following values:

9	4	12	2	6	8	18
---	---	----	---	---	---	----

What is returned by values[3]?

- a. 9
- b. 12
- c. 2
- d. 6
- e. 3

Answer: C

Java array indices start at 0, so values[3] is really the fourth array element, which is 2

Points: 0 / 1

- ☒ 2. For questions 1-2, assume values is an int array that is currently filled to capacity, with the following values:

9	4	12	2	6	8	18
---	---	----	---	---	---	----

What is the value of values.length?

- a. 0
- b. 5
- c. 6
- d. 7
- e. 18

Answer: D

The length operator for an array returns the size of the array. The above picture shows that that values stores 7 elements and since it is full, the size of values is 7.

Points: 0 / 1

- ☒ 3. Which of the following loops would adequately add 1 to each element stored in values?
- a. for(j=1;j<values.length;j++)
values[j]++;
 - b. for(j=0;j<values.length;j++)
values[j]++;
 - c. for(j=0;j<=values.length;j++)
values[j]++;
 - d. for(j=0;j<values.length-1;j++)
values[j]++;
 - e. for(j=1;j<values.length-1;j++)
values[j]++;

Answer: B

The first array element is values[0], so the for-loop must start at 0, not 1. There are values.length elements in the array where the last element is at values.length-1, so the for loop must stop before reaching values.length. This is the case in b. In d, the for loop stops 1 before values.length since "<" is being used to test the condition instead of "<=".

Points: 0 / 1

- ☒ 4. Which of the following is a legal way to declare and instantiate an array of 10 Strings?
- a. String s = new String(10);
 - b. String[10] s = new String;
 - c. String[] s = new String[10];
 - d. String s = new String[10];
 - e. String[] s = new String;

Answer: C

Declaring an array is done by type[] variable. Instantiating the array is done by variable = new type[dimension] where dimension is the size of the array

Points: 0 / 1

- ☒ 5. In Java, arrays are
- a. primitive data types
 - b. objects
 - c. interfaces
 - d. primitive data types if the type stored in the array is a primitive data type and objects if the type stored in the array is an object
 - e. Strings

Answer: B

In Java, arrays are implemented as objects. The variable is a reference variable to the block of memory that stores the entire array. However, arrays are accessed using the notation name[index] rather than by message passing.

Points: 0 / 1

6. The "off-by-one" error associated with arrays arises because
- a. the first array index is 0 and programmers may start at index 1, or may use a loop that goes one index too far
 - b. the last array index is at length + 1 and loops may only iterate to length, missing one
 - c. the last array element ends at length - 1 and loops may go one too far
 - d. programmers write a loop that goes from 0 to length - 1 whereas the array actually goes from 1 to length
 - e. none of the above, the "off-by-one" error has nothing to do with arrays

Answer: A
The array is initialized as `= new type[x]` where `x` is the size of the array. However, the array has legal indices of 0 to `x - 1` and so, programmers are often off-by-one because programmers will write code to try to access indices 1 to `x`.

Points: 0 / 1

7. To declare a two-dimensional int array called `twoD`, which of the following would you use?
- a. `int[2] twoD;`
 - b. `int[,] twoD;`
 - c. `int[[] twoD;`
 - d. `int [[]] twoD;`
 - e. `int[] twoD[2];`

Answer: C
: In Java, you can only declare one-dimensional arrays. To create a two-dimensional array, you must declare it as an array of arrays. The proper notation is to declare the type of array using multiple `[]` marks in succession, as in `int[][]` for a two-dimensional array

Points: 0 / 1

8. What does the following code do?

```
int value1 = Keyboard.readInt();
int value2 = Keyboard.readInt();
bars[value1] += value2;
```

- a. adds 1 to the number of bars sold by child value1 and child value2
- b. adds 1 to the number of bars sold by child value1
- c. adds value1 to the number of bars sold by child value2
- d. adds value2 to the number of bars sold by child value1
- e. inputs a new value for the number of bars sold by both child value1 and child value2

Answer: D
`bars[value1]` is the number of bars sold by child value1, and `+= value2` adds to this value the amount input for value2.

Points: 0 / 1

9. Assume an int array, `candy`, stores the number of candy bars sold by a group of children where `candy[j]` is the number of candy bars sold by child `j`. Assume there are 12 children in all

Which of the following code could be used to compute the total number of bars sold by the children?

- a. `for(int j=0; j<12; j++) sum+= candy[j];`
- b. `for(int j=0; j<12; j++) candy[j] = sum;`
- c. `for(int j=0; j<12; j++) sum = candy[j];`
- d. `for(int j=0; j<12; j++) sum += [j];`
- e. `for(int j=0; j<12; j++) [j] += sum;`

Answer: A
The code in a iterates through all 12 elements of `candy`, adding each value to `sum`. The answer in b sets all 12 elements of `candy` equal to `sum`, the answer in c sets `sum` to be each element of `candy`, resulting in `sum = candy[11]` and d and e have syntactically invalid code

Points: 0 / 1

10. Assume an int array, candy, stores the number of candy bars sold by a group of children where candy[j] is the number of candy bars sold by child j. Assume there are 12 children in all

What does the following method do?

```
public int question15()
{
    int value1 = 0;
    int value2 = 0;
    for(int j=0; j<12; j++)
        if(candy[j] > value1)
        {
            value1 = candy[j];
            value2 = j;
        }
    return value2;
}
```

- a. It returns the total number of candy bars sold
- b. It returns the total number of children who sold 0 candy bars
- c. It returns the total number of children who sold more than 0 candy bars
- d. It returns the number of candy bars sold by the child who sold the most candy bars
- e. It returns the index of the child who sold the most candy bars

Answer: E
The loop iterates through all 12 array elements. If a particular value of candy is found to be larger than value1, then this new value is remembered in value1 along with the index of where it was found in value2. As the loop continues, if a new candy value is found to be greater than the current value1, then it is remembered instead, so the loop finds the maximum number of candy bars sold in value1 and the child's index who sold the most in value2. Since value2 is returned, the code returns the index of the child who sold the most

Points: 0 / 1

11. Consider the array declaration and instantiation: `int[] arr = new int[5];` Which of the following is true about arr?
- a. It stores 5 elements with legal indices between 1 and 5
 - b. It stores 5 elements with legal indices between 0 and 4
 - c. It stores 4 elements with legal indices between 1 and 4
 - d. It stores 6 elements with legal indices between 0 and 5
 - e. It stores 5 elements with legal indices between 0 and 5

Answer: B
Arrays are instantiated with an int value representing their size, or the number of elements that they can store. So, arr can store 5 elements. Further, all arrays start at index 0 and go to index size - 1, so arr has legal indices of 0 through 4.

Points: 0 / 1

12. If an int array is passed as a parameter to a method, which of the following would adequately define the parameter list for the method header?
- a. (int[])
 - b. (int a[])
 - c. (int[] a)
 - d. (int a)
 - e. (a[])

Answer: C
The parameter is defined much as the variable is originally declared, as type parameter name. Here, the type is int[] and the parameter is a

Points: 0 / 1

13. Assume that BankAccount is a predefined class and that the declaration `BankAccount[] firstEmpireBank;` has already been performed. Then the following instruction reserves memory space for

```
firstEmpireBank = new BankAccount[1000];
```

- a. a reference variable to the memory that stores all 1000 BankAccount entries
- b. 1000 reference variables, each of which point to a single BankAccount entry
- c. a single BankAccount entry
- d. 1000 BankAccount entries
- e. 1000 reference variables and 1000 BankAccount entries

Answer: B
The declaration `BankAccount[] firstEmpireBank;` reserves memory space for firstEmpireBank, which itself is a reference variable that points to the BankAccount[] object. The statement `firstEmpireBank = new BankAccount[1000];` instantiates the BankAccount[] object to be 1000 BankAccount objects

Points: 0 / 1

14. The following code accomplishes which of the tasks written below? Assume list is an int array that stores positive int values only.

```
foo = 0;
for(j=0; j<list.length; j++)
    if (list[j] > foo) foo = list[j];
```

- a. it stores the smallest value in list (the minimum) in foo
- b. it stores the largest value in list (the maximum) in foo
- c. it stores every value in list, one at a time, in foo, until the loop terminates
- d. it counts the number of elements in list that are greater than foo
- e. it counts the number of elements in list that are less than foo

Answer: B
The condition in the if statement tests to see if the current element of list is greater than foo. If so, it replaces foo. The end result is that every element in list is tested and foo stores the largest element up to that point, so eventually, foo will be the largest value in the array list.

Points: 0 / 1

15. To initialize a String array names to store the three Strings "Huey", "Duey" and "Louie", you would do

- a. String names = {"Huey", "Duey", "Louie"};
- b. String[] names = {"Huey", "Duey", "Louie"};
- c. String[] names = new String{"Huey", "Duey", "Louie"};
- d. String names[3] = {"Huey", "Duey", "Louie"};
- e. String names; names[0] = "Huey"; names[1] = "Duey"; names[2] = "Louie";

Answer: B
An array does not have to be instantiated with the reserved word new if it is instantiated with the list of values it is to store. So, names = {"Huey", "Duey", "Louie"}; will create a String array of 3 elements with the three values already initialized. Of the other answers, a does not specify that names is a String array, c should not have the reserved word new, d should not have [3] after names and omits [] after String, and e does not instantiate the array as String[3], and thus, all four of these other answers are syntactically invalid

Points: 0 / 1

TRUE/FALSE

16. Arrays have a built in toString method that returns all of the elements in the array as one String with "\n" inserted between each element

Answer: F
Arrays are objects and so they inherit from the Object class. The Object class does have a toString method. However, Object's toString method does not return the value(s) stored in the Object but rather the value of the reference variable. So, toString used on an array will not return the values stored in the array, but instead a meaningless set of characters.

Points: 0 / 1

17. Java arrays can store primitive types and Strings, but cannot store any other type of Object other than Strings.

Answer: F
Java arrays can store any type of class

Points: 0 / 1

18. A Java main method uses the parameter (String[] variable) so that a user can run the program and supply "command-line" parameters. Since the parameter is a String array, however, the user does not have to supply any parameters.


Answer: T
The main method requires the parameter in case a programmer wishes to permit the user to supply command-line parameters. Anything entered at the command line after the java command will then be accepted as the command-line parameter. If it is several words separated by spaces, then each word is stored as a separate String array element. For instance, "java foo.class hi there" would store "hi" in variable[0] and "there" in variable[1] for possible use by the program

Points: 0 / 1

19. An array index cannot be a double, boolean or String.


Answer: T
An array index must be an int type, or a value that can be narrowed into an int (so char, byte and short are also permissible)

Points: 0 / 1

-  20. If the following statement is performed: `CD[] mycollection = new CD[200];` where CD is a previously defined class, then `mycollection[5]` is a CD object.


Answer: T
The variable `mycollection` has been declared as an array of CD objects, so `mycollection[5]` is the 6th CD in the array, and since a CD is an object, `mycollection[5]` is a CD object

Points: 0 / 1

-  21. It is possible to sort an array of `int`, `double` or `String`, but not an array of an `Object` class such as a `CD` class.

Answer: F
It is possible to sort any type of array as long as the type has some mechanism to compare two elements and determine their proper ordering (less than, equal, greater than). So, if the `CD` class has a `compareTo` method, then it would be possible to sort them.

Points: 0 / 1


-  22. To swap the 3rd and 4th elements in the `int` array values, you would do:

```
values[3] = values[4];  
values[4] = values[3];
```

Answer: F
This code first copies `values[4]` into `values[3]` and then copies `values[3]` into `values[4]`. The result is that whatever was stored in `values[4]` is now stored in both `values[3]` and `values[4]`. In order to perform the swap appropriately, you would need a third variable to be used as a temporary storage location, as in


```
int temp = values[3];  
values[3] = values[4];  
values[4] = temp;
```

Points: 0 / 1

-  23. In Java, an array can only store one type of data. For instance, you cannot create an array that stores both `double` and `String` values.


Answer: T
Arrays are known as homogeneous types. This means that the type of value stored in the array must be the same for every element. The type is determined by the declaration. So, `int[] x` makes `x` an array of `int` values only. So, no array could store both `doubles` and `Strings`

Points: 0 / 1

-  24. In a two-dimensional array, both dimensions must have the same number of elements, as in `in[10][10]`.


Answer: F
: In Java, the dimensions can have any number of positive elements, so it is possible to have a two-dimensional array with dimensions of `[5]` and `[10]`.

Points: 0 / 1

-  25. The statement `int[] list = {5, 10, 15, 20};` initializes `list` to have 4 `int` values


Answer: T
An array does not have to be instantiated with the new reserved word if it is instantiated with the list of values it is to store. So, `list = {5, 10, 15, 20};` causes `list` to become an array of 4 `int` values with the values of 5, 10, 15, 20 already initialized. Note that the array is automatically an array of 4 values, so `list[n]` for any value of `n` other than 0, 1, 2 or 3 would result in a thrown `Exception`

Points: 0 / 1

-  26. An array, when instantiated, is fixed in size, but an `ArrayList` can dynamically change in size when new elements are added to it.


Answer: T
: A drawback of an array is its fixed size. Once instantiated, it is fixed in size and so, if a program tries to add more elements than the size of the array, it results in an `Exception` being thrown. The `ArrayList` is a class that uses an array and automatically creates a larger array, copying the old array into the new array so that the `ArrayList` can be larger as needed. While this gets around the problem of a thrown `Exception`, it does lead to poorer performance because of having to copy from one array to another every time the `ArrayList` size is increased

Points: 0 / 1

-  27. If `a` and `b` are both `int` arrays, then `a = b;` will copy all elements of `b` into `a`.


Answer: F
The "=" is an assignment operator. If the two variables are primitives, then the left-hand variable gets a copy of the right-hand variable (so if `a` and `b` are `int` values and `b = 5`, then `a` would become 5). However, since `a` and `b` are arrays, the reference variable `a` is set to the reference variable `b`, resulting in both `a` and `b` referencing the same array in memory, or they are now aliases of each other

Points: 0 / 1

-  28. Unlike an array, an `ArrayList` object can grow and shrink in size

Answer: T
An `ArrayList` is more flexible than an array in this regard


Points: 0 / 1

 29. An ArrayList can store only objects, not primitive types

Answer: T

The add method on the ArrayList class which is used to add elements to the list, takes an Object as a parameter, so only objects can be added to an ArrayList.

Points: 0 / 1

 30. Since a binary search is faster, there is no reason to use a linear search over a binary search.

Answer: F

When the data is not sorted a binary search cannot be used, so a linear search is more appropriate.

Points: 0 / 1