







Name: Jeff Wallace
Date: 10/28/2015
Score: 0 / 26 (0%)

ID: 1018
Class: Mathematics 101

AP Java Chapter 3 Test

TRUE/FALSE


-  1. In Java, if and if-else are selection statements
- Answer:** T
: The if and if-else allow you to make a selection in a program
Points: 0 / 1
-  2. In Java, the symbol "=" and the symbol "==" are used synonymously (interchangeably)
- Answer:** F
Explanation: "=" is used for assignment statements while "==" is used to test equality
Points: 0 / 1
-  3. When comparing any primitive type of variable, == should always be used to test to see if two values are equal
- Answer:** F
This is true of int, short, byte, long, char and boolean, but not of double or float variables. If two double variables, x and y, are being tested, (x == y) is true only if they are exactly equal to the last decimal point. It is common instead to compare these two values but allow for a small difference in value. For instance, if THE TA = 0.000001, we might test x and y by (x - y <= THE TA) instead of (x == y) to get a better idea of if they are close enough to be considered equal.
Points: 0 / 1
-  4. The statements x++; and ++x; will accomplish the same thing
- Answer:** T
The statements x++; and ++x; both increment x
Points: 0 / 1
-  5. The statement { } is a legal block.
- Answer:** T
A block consists of "{", followed by 0 or more Java statements, followed by "}". So it is acceptable to have no statements between the brackets. Situations where this is necessary occur in Java, particularly when implementing methods of abstract classes, something you will study later.
Points: 0 / 1

-  6. The statement `if(a >= b) a++; else b--;` will do the same thing as the statement `if (a < b) b--; else a++;`.

Answer: T

We can reverse the if clause and else clause if we reverse the condition. The opposite condition of `(a >= b)` is `(a < b)` so this works out logically. Note that if we used the condition `(a <= b)` then the resulting statement would not do the same thing as the original `if a==b`


Points: 0 / 1

-  7. An if statement may or may not have an else clause, but an else clause must be part of an if statement

Answer: T

Java allows for either if or if-else statements. But else is only used as part of an if statement

Points: 0 / 1


-  8. The following for-loop is an infinite loop

```
for(int j = 0; j < 1000; ) i++;
```

Answer: T

This loop initializes `j` to 0 and compares it to 1000, but does not alter `j` after each loop iteration. In reality, the loop will terminate with a run-time error eventually once `i` becomes too large of a value to store in memory, but logically, this is an infinite loop

Points: 0 / 1


-  9. The following loop is syntactically valid

```
for(int j = 0; j < 1000; j++) j--;
```

Answer: T

The Java compiler does not care that you are incrementing `j` in the loop but decrementing `j` in the loop body. Logically, this loop makes no sense because `j` will continuously be incremented and decremented so that it never reaches 1000, but there is nothing wrong with the loop syntactically.


Points: 0 / 1

 10. In Java, it is possible to create an infinite loop out of while loops, but not for-loops.

Answer: F

It is true that while loops can be infinite loops, but it is also true that Java for-loops can be infinite loops. This is not true in many other programming languages where for-loops have a set starting and ending point, but Java for-loops are far more flexible than most other language's for-loops


Points: 0 / 1

 11. Software design is the first step in program development.

Answer: F

Before a design can be made we must understand the problem, which is done by establishing requirements.


Points: 0 / 1

 12. The expression 'Z' < 'a' is true.

Answer: T

When characters are compared, their corresponding Unicode value is used, and all uppercase letters have lower Unicode values than all lowercase letters

Points: 0 / 1


 13. Any for loop can be written as a while loop.

Answer: T

In fact, any kind of loop can be written as any other kind of loop. (Note that the if statements are not loops – they are selection statements.) See figure 3.11 for the general equivalent structure of for and while loops

Points: 0 / 1

MULTIPLE CHOICE

 14. During program development, software requirements specify

- a. how the program will accomplish the task
- b. what the task is that the program must perform
- c. how to divide the task into subtasks
- d. how to test the program when it is done
- e. all of the above

Answer: B

The specification phase is to understand the problem at hand so that the programmer can determine what needs to be done to solve the problem. The other efforts listed above are part of the design phase (a, c) and testing phase (d)

Points: 0 / 1

15. If a programmer follows the four phases of program development as intended, which of the four phases should require the least amount of creativity?
- a. Software requirements
 - b. Software design
 - c. Software implementation
 - d. Software testing
 - e. None of the above, all four levels would require equal creativity

Answer: C

Once the implementation phase has been reached, the algorithm should have already been specified, so the only effort involved in the implementation phase is of translating from the design (which is probably in an English-like pseudocode) to the programming language, and entering the code through an editor. The requirements and design phases require understanding the problem and coming up with a solution respectively, requiring creativity, and the testing phase will require diagnostic abilities usually forcing the programmer(s) to be creative in how the errors are found and fixed

Points: 0 / 1

16. Of the following if statements, which one correctly executes three instructions if the condition is true?
- a.

```
if (x < 0)
    a = b * 2;

    y = x;
```
 - b.

```
z = a - y;
{
    if (x < 0)
        a = b * 2;
        y = x;
        z = a - y;
}
```
 - c.


```
if { (x < 0)
    a = b * 2;
    y = x;
    z = a - y;
}
```
 - d.

```
if (x < 0)
{
    a = b * 2;
    y = x;
    z = a - y;
}
```
 - e. b, c and d are all correct, but not a

Answer: D

In order to have three instructions execute in the if clause when the condition is true, the three statements must be placed in a block following the condition. This is the syntax used in d. In a, there is no block. In b, the block is placed around the entire if statement such that the if clause is only `a = b * 2;` and the other two statements are not part of the if statement, but follow it. The syntax in c is illegal resulting in a syntax error. Don't forget that the structure of your code (how it lines up) is immaterial to the compiler

Points: 0 / 1


-  17. Which of the sets of statements below will add 1 to x if x is positive and subtract 1 from x if x is negative but leave x alone if x is 0?

- | | |
|---|---|
| a. <code>if (x > 0) x++;</code>
<code>else x--;</code> | d. <code>if (x == 0) x = 0;</code>
<code>else x++;</code>
<code>x--;</code> |
| b. <code>if (x > 0) x++;</code>
<code>else if (x < 0) x--;</code> | e. <code>x++;</code>
<code>x--;</code> |
| c. <code>if (x > 0) x++;</code>
<code>if (x < 0) x--;</code>
<code>else x = 0;</code> | |

Answer: B

if x is positive, x++ is performed else if x is negative x-- is performed and otherwise, nothing happens, or x is unaffected. In a, c, d and e, the logic is incorrect. In a, x-- is done if x is not positive, thus if x is 0, x becomes -1 which is the wrong answer. In c, if x is positive, then x++ is performed. In either case, the next statement is executed and if x is not negative, the else clause is performed setting x to 0. So if x is positive, it becomes 0 after this set of code. In d, x++ and x-- are both performed if x is not 0. And in e, this code does not attempt to determine if x is positive or negative, it just adds one and then subtracts 1 from x leaving x the same

Points: 0 / 1

-  18. Given the nested if-else structure below, answer question

```

if (a > 0)
    if (b < 0)
        x = x + 5;
    else
        if (a > 5)
            x = x + 4;
        else
            x = x + 3;
else
    x = x + 2;

```

If x is currently 0, a = 5 and b = 5, what will x become after the above statement is executed?

- | | |
|------|------|
| a. 0 | d. 4 |
| b. 2 | e. 5 |
| c. 3 | |

Answer: C

: Since (a > 0) is true, the next condition is checked. Since (b < 0) is false, the else clause for this condition is executed. Since (a > 5) is false the else clause for this condition is executed, which is x = x + 3. Therefore, 3 is added to x, so it is now 3

Points: 0 / 1



19. Given the nested if-else structure below, answer question

```
if (a > 0)
    if (b < 0)
        x = x + 5;
    else
        if (a > 5)
            x = x + 4;
        else
            x = x + 3;
else
    x = x + 2;
```


If x is currently 0, a = 1 and b = -1, what will x become after the above statement is executed?

- | | |
|------|------|
| a. 0 | d. 4 |
| b. 2 | e. 5 |
| c. 3 | |

Answer: E

Since (a > 0) is true, the if clause is executed, which is another if statement. Its condition (b < 0) is true, so its if clause is executed, which is $x = x + 5$, so x is now 5

Points: 0 / 1

 20. What is wrong, logically, with the following code?


```
if (x > 10) System.out.println("Large");  
else if (x > 6 && x <= 10) System.out.println("Medium");  
else if (x > 3 && x <= 6) System.out.println("Small");  
else System.out.println("Very small");
```

- a. There is no logical error, but there is no need to have $(x \leq 10)$ in the second conditional or $(x \leq 6)$ in the third conditional
- b. There is no logical error, but there is no need to have $(x > 6)$ in the second conditional or $(x > 3)$ in the third conditional
- c. The logical error is that no matter what value x is, "Very small" is always printed out
- d. The logical error is that no matter what value x is, "Large" is always printed out
- e. There is nothing wrong with the logic at all

Answer: A

Because this is a nested if-else statement, if $(x > 10)$ is true, then the first println statement is executed and the rest of the statement is skipped. If $(x > 10)$ is not true, then the first else clause is executed and the next if condition is tested. At this point, $(x > 10)$ is known to be false and therefore $(x \leq 10)$ must be true, so there is no need to check this inequality. Similarly, if $(x > 6)$ is false, then the second else clause is executed and the third if condition is tested. However, $(x \leq 6)$ must be true, so there is no need to check this inequality.

Points: 0 / 1

 21. If x is an int where $x = 1$, what will x be after the following loop terminates?


```
while (x < 100)  
    x *= 2;
```

- a. 2
- b. 64
- c. 100
- d. 128
- e. None of the above, this is an infinite loop

Answer: D

With $x = 1$, the loop condition is true and the statement executes, doubling x , which is now 2. Since the loop condition is still true, the statement executes again, doubling x to 4. The condition remains true for the next 4 iterations, where x becomes 8, 16, 32 and 64. Since $(x < 100)$ is still true when $x = 64$, the loop iterates again and x becomes $2 * 64 = 128$. Now, $(x < 100)$ is no longer true and the loop terminates with $x = 128$.

Points: 0 / 1

-  22. Given the following code, where $x = 0$, what is the resulting value of x after the for-loop terminates?


```
for(int i=0;i<5;i++)  
    x += i;
```

- | | |
|------|-------|
| a. 0 | d. 10 |
| b. 4 | e. 15 |
| c. 5 | |

Answer: D

Each pass through the for-loop results with the current value of the loop index, i , being added to x . The first time through the loop, $i = 0$ so $x = x + 0 = 0$. The second time through the loop $i = 1$ so $x = x + 1 = 1$. The third time through the loop $i = 2$ so $x = x + 2 = 3$. The fourth time through the loop $i = 3$ so $x = x + 3 = 6$. The fifth and final time through the loop $i = 4$ so $x = x + 4 = 10$

Points: 0 / 1

-  23. How many times will the following loop iterate?


```
int x = 10;  
while (x > 0)  
{  
    System.out.println(x);  
    x--;
```

- | | |
|------------|-------------|
| a. 0 times | d. 10 times |
| b. 1 time | e. 11 times |
| c. 9 times | |

Answer: D

Since the condition is tested before the loop body executes, the loop will not execute once $x == 0$. So, the loop iterates for $x = 10, x = 9, \dots, x = 1$, or 10 times


Points: 0 / 1

-  24. Given two String variables, s1 and s2, to determine if they are the same length, which of the following conditions would you use?
- a. `(s1.equals(s2))`
 - b. `(s1.length().equals(s2))`
 - c. `(s1.length().equals(s2.length()))`
 - d. `(s1.length() == s2.length())`
 - e. `length(s1) == length(s2)`

Answer: D

Since s1 and s2 are Strings, we can only obtain their length by passing the `length()` message to each one. The `length()` method returns an int that we can directly compare against another int using `==`, as done in d. The answers for b and c are syntactically invalid since `s1.length()` returns an int, and so cannot be passed the message `.equals()`. The answer in e is also syntactically invalid. Finally, the answer in a determines if the two Strings are the exact same. While their lengths will be equal if this is true, two Strings may have equal length but not be equal, so this is logically incorrect.

Points: 0 / 1

-  25. Given that s is a String, what does the following loop do?

```
for(int j = s.length(); j > 0; j--)  
    System.out.print(s.charAt(j-1));
```

- a. it prints s out backwards
- b. it prints s out forwards
- c. it prints s out backwards after skipping the last character
- d. it prints s out backwards but does not print the 0th character
- e. it yields a run-time error because there is no character at `s.charAt(j-1)` for `j = 0`

Answer: A

The variable j counts down from the length of the String to 1, each time printing out the character at position j-1. The character at length - 1 is the first character printed and this is the last character of the String. It continues until it reaches j = 1, and prints out the character at position j - 1, or the 0th character, so it prints the entire String backwards.

Points: 0 / 1



26. The following nested loop structure will execute the inner most statement (x++) how many times?

```
for(int j = 0; j < 100; j++)  
    for(int k = 100; k > 0; k--)  
        x++;
```

- a. 100
- b. 200
- c. 10,000
- d. 20,000
- e. 1,000,000

Answer: C

The outer loop iterates 100 times. Each time it iterates, the inner loop, and the x++ statement, execute 100 times. Therefore, the statement x++ executes $100 \times 100 = 10,000$ times

Points: 0 / 1