socrative

Answer Key
February 24, 2021

Java OCA Chapter 2 Review ('Copy')

1. Which of the Java following operators can be used with boolean variables? (Choose all that apply)

- ✓ **A** ==
- **B** +
- **C** --
- ✓ **D** !
- **E** %
- **F** <=

i Option A is the equality operator and can be used on numeric primitives, boolean values, and object references. Options B and C are both arithmetic operators and cannot be applied to a boolean value. Option D is the logical negation operator and is used exclusively with boolean values. Option E is the modulus operator, which can only be used with numeric primitives. Finally, option F is a relational operator used that compares the values of two numbers.

2. What data type (or types) will allow the following code snippet to compile? (Choose all that apply) byte x = 5; byte y = 10; _____ z = x + y;

- ✓ **A** int
- ✓ **B** long
- **C** boolean
- ✓ **D** double
- **E** short
- **F** byte

i The value is x + y is automatically promoted to int, so int and data types that can be promoted automatically from int will work. Options A, B, D are such data types. Option C will not work because boolean is not a numeric data type. Options E and F will not work without an explicit cast to a smaller data type.

3. What is the output of the following application? 1: public class CompareValues { 2: public static void main(String[] args) { 3: int x = 0; 4: while(x++ < 10) {} 5: String message = x > 10 ? "Greater than" : false; 6: System.out.println(message+","+x); 7: } 8: }

- A Greater than,10
- B false,10
- C Greater than,11
- D false,11
- E Does not compile because of line 4
- ✓ F Does not compile because of line 5

i In this example, the ternary operator has two expressions, one of them a String and the other a boolean value. The ternary operator is permitted to have expressions that don't have matching types, but the key is here the assignment to the String reference. The compiler knows how to assign the first expression value as a String, but the second boolean expression cannot be set as a String; therefore, this line will not compile.

4. What change would allow the following code snippet to compile? (Choose all that apply) 3: long x = 10; 4: int y = 2 * x;

- A No change; it compiles as is.
- ✓ B Cast x on line 4 to int.
- ✓ C Change the data type of x on line 3 to short.
- ✓ D Cast 2 * x on line 4 to int.
- E Change the data type of y on line 4 to short.
- ✓ F Change the data type of y on line 4 to long.

i The code will not compile as is, so option A is not correct. The value 2 * x is automatically promoted to long and cannot be automatically stored in y, which is in an int value. Options B, C, and D solve this problem by reducing the long value to int. Option E does not solve the problem and actually makes it worse by attempting to place the value in a smaller data type. Option F solves the problem by increasing the data type of the assignment so that long is allowed.

5. What is the output of the following code snippet? 3: java.util.List<Integer> list = new java.util.ArrayList<Integer>(); 4: list.add(10); 5: list.add(14); 6: for(int x : list) { 7: System.out.print(x + ", "); 8: break; 9: }

- A 10, 14,
- B 10, 14
- ✓ C 10,
- D Does not compile because of line 7
- E Does not compile because of line 8
- F The code contains an infinite loop and does not terminate.

i This code does not contain any compilation errors or an infinite loop, so options D, E, and F are incorrect. The break statement on line 8 causes the loop to execute once and finish, so option C is the correct

answer.

6. What is the output of the following code snippet? 3: int x = 4; 4: long y = x * 4 - x++; 5: if(y<10) System.out.println("Too Low"); 6: else System.out.println("Just right"); 7: else System.out.println("Too High");

- A Too Low
- B Just Right
- C Too High
- D Compiles but throws a NullPointerException
- E Does not compile because of line 6
- ✓ F Does not compile because of line 7

i The code does not compile because two else statements cannot be chained together without addition if-then statements, so the correct answer is option F. Option E is incorrect as Line 6 by itself does not cause a problem, only when it is paired with Line 7. One way to fix this code would be to add an if-then statement on line 6. The other solution would be to remove line 7.

7. What is the output of the following code? 1: public class TernaryTester { 2: public static void main(String[] args) { 3: int x = 5; 4: System.out.println(x > 2 ? x < 4 ? 10 : 8 : 7); 5: }}

- A 5
- B 4
- C 10
- ✓ D 8
- E 7
- F Does not compile because of line 4

i As you learned in the section “Ternary Operator,” although parentheses are not required, they do greatly increase code readability, such as the following equivalent statement: System.out.println((x > 2) ? ((x < 4) ? 10 : 8) : 7) If we work backward, we can reduce the expression by applying the inner ternary statement first, reducing the problem to: System.out.println((x > 2) ? 8 : 7) Since x is greater than 2, the answer is 8, or option D in this case.

8. What is the output of the following code snippet? 3: boolean x = true, z = true; 4: int y = 20; 5: x = (y != 10) ^ (z=false); 6: System.out.println(x+", "+y+", "+z);

- A true, 10, true
- ✓ B true, 20, false
- C false, 20, true
- D false, 20, false
- E false, 20, true
- F Does not compile because of line 5

i This example is tricky because of the second assignment operator embedded in line 5. The expression

(z=false) assigns the value false to z and returns false for the entire expression. Since y does not equal 10, the left-hand side returns true; therefore, the exclusive or (^) of the entire expression assigned to x is true. The output reflects these assignments, with no change to y, so option B is the only correct answer. The code compiles and runs without issue, so option F is not correct.

9. How many times will the following code print "Hello World"? 3: for(int i=0; i<10 ;) { 4: i = i++; 5: System.out.println("Hello World"); 6: }

- A 9
- B 10
- C 11
- D The code will not compile because of line 3.
- E The code will not compile because of line 5.
- ✓ F The code contains an infinite loop and does not terminate.

i In this example, the update statement of the for loop is missing, which is fine as the statement is optional, so option D is incorrect. The expression inside the loop increments i but then assigns i the old value. Therefore, i ends the loop with the same value that it starts with: 0. The loop will repeat infinitely, outputting the same statement over and over again because i remains 0 after every iteration of the loop.

10. What is the output of the following code? 3: byte a = 40, b = 50; 4: byte sum = (byte) a + b; 5: System.out.println(sum);

- A 40
- B 50
- C 90
- ✓ D The code will not compile because of line 4.
- E An undefined value

i Line 4 generates a possible loss of precision compiler error. The cast operator has the highest precedence, so it is evaluated first, casting a to a byte. Then, the addition is evaluated, causing both a and b to be promoted to int values. The value 90 is an int and cannot be assigned to the byte sum without an explicit cast, so the code does not compile. The code could be corrected with parentheses around (a + b), in which case option C would be the correct answer.

11. What is the output of the following code? 1: public class ArithmeticSample { 2: public static void main(String[] args) { 3: int x = 5 * 4 % 3; 4: System.out.println(x); 5: }}

- ✓ A 2
- B 3
- C 5
- D 6
- E The code will not compile because of line 3.

i The * and % have the same operator precedence, so the expression is evaluated from left-to-right. The result of 5 * 4 is 20, and 20 % 3 is 2 (20 divided by 3 is 18, the remainder is 2). The output is 2 and option

A is the correct answer.

12. What is the output of the following code snippet? 3: int x = 0; 4: String s = null; 5: if(x == s) System.out.println("Success"); 6: else System.out.println("Failure");

- A Success
- B Failure
- C The code will not compile because of line 4.
- ✓ D The code will not compile because of line 5.

i The variable x is an int and s is a reference to a String object. The two data types are incomparable because neither variable can be converted to the other variable's type. The compiler error occurs on line 5 when the comparison is attempted, so the answer is option D.

13. What is the output of the following code snippet? 3: int x1 = 50, x2 = 75; 4: boolean b = x1 >= x2; 5: if(b = true) System.out.println("Success"); 6: else System.out.println("Failure");

- ✓ A Success
- B Failure
- C The code will not compile because of line 4.
- D The code will not compile because of line 5.

i The code compiles successfully, so options C and D are incorrect. The value of b after line 4 is false. However, the if-then statement on line 5 contains an assignment, not a comparison. The variable b is assigned true on line 3, and the assignment operator returns true, so line 5 executes and displays Success, so the answer is option A.

14. What is the output of the following code snippet? 3: int c = 7; 4: int result = 4; 5: result += ++c; 6: System.out.println(result);

- A 8
- B 11
- ✓ C 12
- D 15
- E 16
- F The code will not compile because of line 5.

i The code compiles successfully, so option F is incorrect. On line 5, the pre-increment operator is used, so c is incremented to 4 and the new value is returned to the expression. The value of result is computed by adding 4 to the original value of 8, resulting in a new value of 12, which is output on line 6. Therefore, option C is the correct answer.

15. What is the output of the following code snippet? 3: int x = 1, y = 15; 4: while x < 10 5: y—; 6: x++; 7: System.out.println(x+", "+y);

A 10, 5

B 10, 6

C 11, 5

D The code will not compile because of line 3.

✓ E The code will not compile because of line 4.

F The code contains an infinite loop and does not terminate.

i This is actually a much simpler problem than it appears to be. The while statement line 4 is missing parentheses, so the code will not compile, and option E is the correct answer. If the parentheses were added, though, option F would be the correct answer since the loop does not use curly braces to include x++ and the boolean expression never changes. Finally, if curly braces were added around both expressions, the output would be 10, 6 and option B would be correct.

16. What is the output of the following code snippet? 3: do { 4: int y = 1; 5: System.out.print(y++ + " "); 6: } while(y <= 10);

A 1 2 3 4 5 6 7 8 9

B 1 2 3 4 5 6 7 8 9 10

C 1 2 3 4 5 6 7 8 9 10 11

✓ D The code will not compile because of line 6.

E The code contains an infinite loop and does not terminate.

i The variable y is declared within the body of the do-while statement, so it is out of scope on line 6. Line 6 generates a compiler error, so option D is the correct answer.

17. What is the output of the following code snippet? 3: boolean keepGoing = true; 4: int result = 15, i = 10; 5: do { 6: i--; 7: if(i==8) keepGoing = false; 8: result -= 2; 9: } while(keepGoing); 10: System.out.println(result);

A 7

B 9

C 10

✓ D 11

E 15

F The code will not compile because of line 8.

i The code compiles without issue, so option F is incorrect. After the first execution of the loop, i is decremented to 9 and result to 13. Since i is not 8, keepGoing is false, and the loop continues. On the next iteration, i is decremented to 8 and result to 11. On the second execution, i does equal 8, so keepGoing is set to false. At the conclusion of the loop, the loop terminates since keepGoing is no longer true. The value of result is 11, and the correct answer is option D.

18. What is the output of the following code snippet? 3: int count = 0; 4: ROW_LOOP: for(int row = 1; row <=3; row++) 5: for(int col = 1; col <=2 ; col++) { 6: if(row * col % 2 == 0) continue ROW_LOOP; 7: count++; 8: } 9: System.out.println(count);

A 1

✓ B 2

C 3

D 4

E 6

F The code will not compile because of line 6.

i The expression on line 5 is true when row * col is an even number. On the first iteration, row = 1 and col = 1, so the expression on line 6 is false, the continue is skipped, and count is incremented to 1. On the second iteration, row = 1 and col = 2, so the expression on line 6 is true and the continue ends the outer loop with count still at 1. On the third iteration, row = 2 and col = 1, so the expression on line 6 is true and the continue ends the outer loop with count still at 1. On the fourth iteration, row = 3 and col = 1, so the expression on line 6 is false, the continue is skipped, and count is incremented to 2. Finally, on the fifth and final iteration, row = 3 and col = 2, so the expression on line 6 is true and the continue ends the outer loop with count still at 2. The result of 2 is displayed, so the answer is option B.

19. What is the result of the following code snippet? 3: int m = 9, n = 1, x = 0; 4: while(m > n) { 5: m--; 6: n += 2; 7: x += m + n; 8: } 9: System.out.println(x);

A 11

B 13

C 23

✓ D 36

E 50

F The code will not compile because of line 7.

i Prior to the first iteration, m = 9, n = 1, and x = 0. After the iteration of the first loop, m is updated to 8, n to 3, and x to the sum of the new values for m + n, 0 + 11 = 11. After the iteration of the second loop, m is updated to 7, n to 5, and x to the sum of the new values for m + n, 11 + 12 = 23. After the iteration of the third loop, m is updated to 6, n to 7, and x to the sum of the new values for m + n, 23 + 13 = 36. On the fourth iteration of the loop, m > n evaluates to false, as 6 < 7 is not true. The loop ends and the most recent value of x, 35, is output, so the correct answer is option D.

20. What is the result of the following code snippet? 3: final char a = 'A', d = 'D'; 4: char grade = 'B'; 5: switch(grade) { 6: case a: 7: case 'B': System.out.print("great"); 8: case 'C': System.out.print("good"); break; 9: case d: 10: case 'F': System.out.print("not good"); 11: }

A great

✓ B greatgood

C The code will not compile because of line 3.

D The code will not compile because of line 6.

E The code will not compile because of lines 6 and 9.

The code compiles and runs without issue, so options C, D, and E are not correct. The value of grade is 'B' and there is a matching case statement that will cause "great" to be printed. There is no break statement after the case, though, so the next case statement will be reached, and "good" will be printed. There is a break after this case statement, though, so the switch statement will end. The correct answer is thus option B.